**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**


# TRABAJO FIN DE GRADO

# Estudio de mecanismos de autenticación basados en contraseñas visuales en dispositivos móviles Android

**Mario González Nahón**

**Mayo 2014**

# Estudio de mecanismos de autenticación basados en contraseñas visuales en dispositivos móviles Android

# (COSITI_3/1314)

**AUTOR: Mario González Nahón**

**TUTOR: David Arroyo Guardeño**

**Ponente: Gonzalo Martínez Muñoz**

**Universidad Autónoma de Madrid**

**Escuela Politécnica Superior**

**Mayo 2014**

# Agradecimientos

A mi tutor, David, por proponerme este Trabajo Fin de Grado y ayudarme con el mismo siempre que lo he necesitado.

A mis compañeros de universidad, en especial a Cristina por ser la mejor compañera que haya podido tener y por todo el apoyo que me ha dado siempre. Y también a Álvaro por su ayuda en este trabajo.

A mis compañeros de trabajo en Mercanza por darme el tiempo necesario para poder realizar este trabajo.

A mis padres y mi familia por estar siempre a mi lado y por todo lo que me habéis dado siempre. Ya que sin ellos no sería quien soy ni estaría donde estoy.

A todos muchas gracias.

# Resumen

Todo sistema de seguridad de la información debe cumplir con una serie de requisitos mínimos de confidencialidad, integridad y disponibilidad para que pueda considerarse realmente seguro y fiable. Ahora bien, en el estado actual de las Tecnologías de la Información y de la Comunicación también es necesario implantar sistemas de control de accesos a los recursos compartidos a través de sistemas distribuidos. En este sentido a los requerimientos básicos hemos de añadir los objetivos de autenticación y autorización.

Por otro lado, una máxima fundamental en el despliegue de cualquier tecnología es el grado de aceptación por parte de sus potenciales receptores. Así, en el campo específico de los Sistemas de Gestión de Seguridad de la Información no hay cabida para esquemas que promuevan la consecución de objetivos de seguridad a costa de dificultar su implementación y/o uso. En este trabajo el foco central vendrá determinado por el estudio de sistemas de autenticación que, consecuentemente, traten de conciliar el objetivo de seguridad de validación de identidades digitales con la fácil adopción por parte de desarrolladores y usuarios finales.

En efecto, los métodos de autenticación en protocolos de comunicación se pueden medir por dos factores: la usabilidad y la seguridad. En algunos casos cuando se intenta implementar un sistema de autenticación usable se vuelve poco *seguro* y en otros casos que se intenta implementar un sistema de autenticación *seguro* convirtiéndolo en poco usable. El diseño de estos sistemas de autenticación debe ser robusto ya que es un componente crítico de los sistemas de seguridad. La mayoría de los sistemas de autenticación utilizan un sistema basado en contraseñas alfanuméricas y en la mayoría de estos casos es el propio usuario quien, por usabilidad, elige su contraseña. Esta posibilidad que se da al usuario para elegir la contraseña puede ser una debilidad, ya que las contraseñas más utilizadas en estos casos son fechas de cumpleaños o nombres de familiares o amigos.

Nuestra propuesta para mantener la seguridad y a la vez tener una alta usabilidad pasa por cambiar las contraseñas alfanuméricas por contraseñas visuales. Las contraseñas visuales ofrecen una serie de ventajas frente a las contraseñas alfanuméricas que las hacen más seguras y usables. A lo largo de este trabajo se pone de relieve el conjunto de ventajas, así como las limitaciones de esquemas de autenticación basados en las denominadas contraseñas visuales.

Las contraseñas visuales determinan una interacción con el usuario mediada por una o varias imágenes. La autenticación del usuario viene codificada mediante la selección por parte del mismo de una serie de elementos o propiedades de una o varias imágenes. Dada la modalidad de la interacción usuario-imágenes parece recomendable trabajar con interfaces que agilicen la captura de entradas de usuario. Éste es el caso de las pantallas táctiles, la interfaz usuario-máquina cuya popularidad se ha visto favorecido por la reducción de costes de producción y cuya hegemonía en las interfaces de usuario se ha fraguado debido a su inclusión en la telefonía móvil. En este Trabajo Fin de Grado, por ello, se estudiarán los sistemas de identificación de usuarios arriba señalados tomando como marco de implementación los *teléfonos inteligentes*. De modo más concreto, se trabajará con dispositivos móviles con sistema operativo Android, siendo la razón fundamental de esta elección la popularidad de

dichos dispositivos y la facilidad de desarrollo de nuevas aplicaciones frente a otras alternativas.

**Palabras clave:** Confidencialidad, integridad, disponibilidad, sistema de autenticación, contraseña visual, usabilidad, seguridad, Android.

# Abstract

Secure and reliable information systems are designed to preserve confidentiality, integrity and availability. In addition, in the current state of Information and Communication Technologies it is also necessary to develop access control systems to manage shared resources in distributed systems. Consequently, we have to enlarge the original set of requirements by means of the authentication and authorization commitments.

On the other hand, a fundamental target in the deployment of any technology is the degree of acceptance by its potential users. In Information Security Management Systems there is no place for schemes obtaining security by hindering its implementation and / or use. In this work the focus will be determined by the study of authentication systems that enable an adequate tradeoff between secure digital identities management and easy-to-deploy/easy-to use implementations.

In fact, the authentication methods in communication protocols can be measured by two factors: usability and security. In some case when trying to implement a usable authentication system it becomes insecure an in other cases when trying to implement a secure authentication it becomes unusable. The design of these authentication systems has to be robust, because it is a critical component of security systems. Most of the authentication systems use a system based on alphanumeric passwords and in most of these cases is the user who chooses her/his password. This user capacity to choose the password can be a weakness, as the most frequently used password in these cases are birthdays acquaintance names.

Our approach to keep security while to have a high usability depends on changing the alphanumeric passwords for graphical passwords. Graphical passwords offer a number of advantages over alphanumeric passwords that make them more secure and usable. Along this work we highlight the set of advantages and limitations of authentication schemes based on the so-called graphical passwords.

Graphical passwords are built upon an image-based human-machine interface. User authentication is codified by the user selection on a given set of elements or properties of one or several images. Taken into account the modality of the user-image interaction, it is necessary to choose human-machine interfaces that make easy to capture and process the related user input. This target is achieved through touch-screens and their user-machine interfaces, whose popularity sustained by the decrements of production costs, and its incorporation as main user-machine interface in mobile devices. This being the use, in this project we study in detail authentication procedures in the context of smartphones. More specifically, we will work with mobile devices with Android operating system, being the main reason for this choice the popularity of these devices and their ease of development of new applications over other alternatives.

**Keyword:** confidentiality, integrity, availability, authentication system, visual password, usability, security, Android.

# Índice de contenido

# Índice de ilustraciones



# Índice de tablas

# 1. Introducción

## 1.1 Motivación

El surgimiento de los primeros sistemas de escritura y su posterior perfeccionamiento ha ido acompañado de una creciente preocupación al respecto de los potenciales receptores de la información involucrada. En determinados contextos el emisor o generador de la información desea que ésta solo puede ser recibida por un conjunto de receptores.

Ya en el siglo V a.C. los griegos inventaron un instrumento para cifrar mensajes. Este instrumento se llamaba *Escítala Lacedemonia* y consistía en un cilindro de madera en el cual se enrollaba una cinta de papiro [1]. El mensaje se escribía a lo largo de las generatrices del cilindro, después lo desenrollaban y lo enviaban. De esta forma el mensaje solo podía ser leído si el receptor tenía un cilindro del mismo tamaño. Si avanzamos hasta la edad Media, Roger Bacon (1214-1294) expone en una de sus obras los conocimientos sobre criptografía que se tenían hasta entonces, estas técnicas se basaban en:

- Reemplazar unas letras por otras.
- Escribir palabras al revés.
- Invertir letras alternadas en el texto del mensaje original.
- Sustituir las letras pos símbolos.
- Reemplazar cada letra con otras de forma que las suba de sus valores numéricos fuera igual al valor de la letra reemplazada.
- Sustituir cada letra con el nombre de una persona.
- Sustituir las letras por signos.

El conjunto de procedimientos de sustitución y permutación clásicos fueron progresivamente mejorados, hasta que ya en el siglo XX Claude Shannon enunció los principios de la seguridad (teórica) perfecta [2]. Sin embargo, todas estas técnicas únicamente buscaban la confidencialidad del mensaje.

Con el paso del tiempo y la evolución de los sistemas de información y la automatización de los mismos, han entrado en juego otros factores como la integridad del mensaje y su disponibilidad. Así mismo, la capacidad de relacionar una entidad física con una identidad digital es de gran importancia, ya que con ello conseguimos otro factor muy importante que es la autenticación. Pero la autenticación por sí sola no es muy útil ya que es necesario una autorización para tener acceso a los activos de la información. La unión de la autenticación y autorización nos permite crear los sistemas de autenticación.

Los sistemas de autenticación son una de las partes más importantes en los vigentes sistemas de seguridad de la información. Por ello el diseño de estos sistemas debe ser robusto ya que es un componente crítico. Entre la variedad de mecanismos posibles de autenticación, los procedimientos basados en contraseñas alfanuméricas son una de las opciones más ampliamente adoptada. En la mayoría de casos es el propio usuario quien elige su contraseña. Este factor humano es considerado como la parte más débil de los sistemas de autenticación ya que deja a elección del usuario la contraseña, lo que puede conducir a contraseñas débiles

como pueda ser una fecha o un nombre de pila o también contraseñas vulnerables a ataques por diccionario. Es cierto que estos procedimientos de validación de usuarios son de fácil configuración y uso por parte del usuario final, pues de forma general es el usuario quien elige la contraseña que puede recordar de forma sencilla. Sin embargo y pese a que la usabilidad de estos sistemas es un punto importante, no se debe descuidar el requerimiento básico de seguridad, esto es, la dificultad por parte de un tercer usuario de eludir el sistema de autenticación establecido.

## 1.2 Objetivos

En este punto se propone un cambio en este paradigma de contraseñas alfanuméricas para adoptar un sistema basado en contraseñas visuales. Las contraseñas visuales ofrecen como ventajas que el espacio de claves puede ser muy superior y que además son más resistentes a los ataques convencionales contra los sistemas de autenticación basados en contraseñas. Además tienen a su favor que recordar contraseñas basadas en imágenes resulta más sencillo que recodar una palabra no incluida en un diccionario. Con este sistema queremos conseguir una alta usabilidad sin perjudicar a la seguridad.

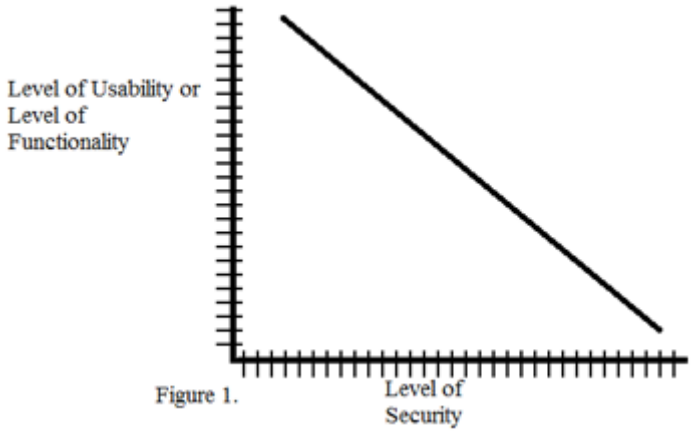


*Ilustración 1: Relación entre Usabilidad y Seguridad*

La usabilidad también depende en muchos casos de la tecnología utilizada y en la actualidad el mundo de las tecnologías ha evolucionado muy rápido en los últimos años. Una de estas tecnologías que más se ha integrado en nuestras vidas son los teléfonos inteligentes o *smartphones* y *tablets*. Estos dispositivos unen funcionalidades de un teléfono y de un ordenador, de forma que podemos ejecutar en ellos aplicaciones para trabajar, entretenernos o acceder a páginas web o el correo electrónico, del mismo modo que podemos llamar o enviar y recibir mensajes de texto. Además la interacción con estos dispositivos se realiza mediante una pantalla táctil ya que carecen de teclado físico y ratón. Entre todos los dispositivos podemos destacar aquellos que implementan Android como sistema operativo, ya que son los más populares debido a su bajo coste y a que es de código abierto u o*pen source.*

De esta forma, el principal objetivo de este proyecto es la de realizar un estudio sobre las ventajas e inconvenientes que pueden tener los sistemas de autenticación basados en contraseñas visuales frente a otros sistemas en dispositivos móviles Android.

A parte del trabajo de estudio y documentación, se desarrollará una aplicación para Android donde se implementen varios sistemas de autenticación para entender mejor las ventajas e inconvenientes de estos sistemas.

## 1.3 Esquema

Una vez aclaradas nuestras intenciones y objetivos en la realización de este proyecto, analizaremos en profundidad los aspectos que hemos explicado en la visión general del proyecto, así como la solución presentada.

Empezaremos por la definición y características de los sistemas de seguridad de la información en el capítulo 2. Dentro de los sistemas de seguridad, explicaremos con más detalle los sistemas de autenticación, sus características y problemas que existen actualmente. En este capítulo además explicaremos la solución propuesta para mejorar la usabilidad de estos sistemas manteniendo un nivel alto en la seguridad mediante los sistemas de autenticación basado en contraseñas visuales. Donde explicaremos con detalle en qué consisten, su clasificación y los problemas que existen con estos sistemas.

A continuación en el capítulo 3, destacamos y utilizamos las ventajas de los dispositivos móviles Android a la hora de conseguir parte de la usabilidad buscada. A este respecto se subrayan las características que han hecho de Android el sistema más popular en los dispositivos móviles, pero al mismo tiempo hacemos énfasis en algunos de los problemas derivados de escoger dicha plataforma como base de nuestras implementaciones.

En el capítulo 4 realizamos el análisis y la implantación de la solución que hemos presentado. Describiremos el proyecto comentado las herramientas utilizadas y describiendo los requisitos funcionales y no funcionales que se deben conseguir. Mostraremos y explicaremos la estructura del proyecto. Seguidamente explicaremos los métodos implementados y cómo los hemos implementado. Y por último en este capítulo veremos cómo se han realizado las pruebas, las herramientas utilizadas y los resultados obtenidos.

En el capítulo 5 terminamos el trabajo con las conclusiones a las hemos llegado después de todo el trabajo anterior y las competencias adquiridas. Asimismo discutiremos las posibles líneas del trabajo futuro e implicaciones que tendría este proyecto.

# 2. Sistemas de seguridad de la información

En este capítulo hablaremos sobre los sistemas de seguridad y las características que deben cumplir para ser tipificados como *seguros*. También comentaremos una de las partes más críticas de los sistemas de seguridad, los sistemas de autenticación, los retos que plantean y las soluciones estándar que han sido propuestas.

## 2.1 Introducción a los sistemas de seguridad de la información

Los algoritmos de cifrado actuales como el AES o el DES [3], sirven para evitar que un tercero pueda leer la información que protegen, ya que mediante un ataque de fuerza bruta llevaría mucho tiempo encontrar la clave y para cuando se encontrase la clave, la información ya no sería válida. Pero la *seguridad* no se basa únicamente en cifrar la información, tiene como objetivo la protección de la información y de los sistemas de la información del acceso, uso, divulgación interrupción o destrucción no autorizada. La seguridad es un concepto asociado a la certeza, falta de riesgo o contingencia. Podemos entender como seguridad un estado de cualquier sistema o tipo de información que nos indica que ese sistema o información está libre de peligro, daño o riesgo. Se entiende como peligro o daño todo aquello que pueda afectar a su funcionamiento directo o a los resultados que se obtienen. Hay que aclarar que la seguridad absoluta no es posible y no existe un sistema 100% seguro, de forma que el riesgo siempre está presente. Así, en la actualidad la definición de seguridad perfecta de Shannon se modula de acuerdo con las expectativas de los usuarios y las amenazas de las tecnologías implicadas. Esta transición lleva a definir la seguridad de la información en base a objetivos más específicos y concretos.

La seguridad de la información es el conjunto de medidas de organizaciones y sistemas tecnológicos que permiten proteger la información teniendo como objetivo mantener la confidencialidad, la disponibilidad e integridad de la misma. Estos son los tres principios básicos a los que debe aspirar todo sistema de seguridad de la información, conocidos como **Triada CID.** [4]


*Ilustración 2: Triada CID*

### 2.1.1 Confidencialidad

La confidencialidad es la propiedad que impide la divulgación de información a terceros no autorizados. Asegura el acceso a la información solo a aquellas personas que tengan autorización. Entre las principales amenazas de la confidencialidad están:

- Monitorizar el tráfico de la red
- *Keylogger* para capturar el teclado o la pantalla
- Robo de ficheros de contraseñas
- La ingeniería social para averiguar contraseñas.

Como contramedidas se encuentran:

- Cifrar los datos tanto los almacenados como los que se envían
- Implementando un control de acceso estricto y clasificando la información por usuarios
- Enseñando al personal los procedimientos adecuados de seguridad para elegir contraseñas o proteger archivos.

### 2.1.2 Integridad

La integridad es la propiedad que busca mantener los datos intactos, libres de modificaciones no autorizadas. Así mismo hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado y que esta modificación sea registrada, asegurando su precisión y confiabilidad. Entre las principales causas del fallo en la integridad están:

- Virus
- Bombas lógicas
- Uso de puertas traseras de acceso

Y como contramedidas:

- Implementando un control de acceso estricto
- Sistemas de detección de intrusos
- Firma digital

### 2.1.3 Disponibilidad

La disponibilidad es la propiedad de la información de encontrarse a disposición de quienes deban acceder a ella, ya sean personas, procesos o aplicaciones. La disponibilidad es el acceso a la información y a los sistemas por personas autorizadas siempre que quieran. Las amenazas que puedan entorpecer la disponibilidad son:

- Fallo en los sistemas
- Mal mantenimiento de los equipos o malas condiciones ambientales que puedan provocar fallos en los servicios
- Ataques de denegación de servicio (en inglés, *Denial of Service –DoS-*)

Y entre los mecanismos para cumplir con los niveles de disponibilidad se encuentran:

- Mantenimiento de sistemas de respaldo (*backups)* de los activos de información para reemplazar los sistemas fallidos
- Monitorización del tráfico de red y del sistema anfitrión
- Uso de ciertas configuraciones de cortafuegos y *routers*

### 2.1.4 Otros factores

Además de los principios fundamentales, anteriormente nombrados, hay otros factores importantes a tener en consideración dentro de un sistema de seguridad de la información.

#### *2.1.4.1 No Repudio*

El no repudio es la propiedad que permite asegurar el destino o el origen de una información. El servicio de Seguridad de No Repudio está estandarizado en la ISO-7498-2 [5].

El no repudio de origen implica que el emisor no pueda negar el envío ya que el receptor tiene pruebas infalsificables del origen del envío, lo cual evita que el emisor pueda negar el envío de la información. El emisor es el encargado de generar esta prueba que se envía al receptor.

El no repudio de destino implica que el receptor no pueda negar la recepción de la información ante el emisor. En este caso la prueba la genera el receptor y la envía al emisor.

El no repudio evita que el emisor o el receptor nieguen la transmisión de un mensaje. Así, cuando se envía un mensaje, el receptor puede comprobar que el supuesto emisor envió el mensaje. De la misma forma, cuando se recibe un mensaje, el emisor puede verificar que el supuesto receptor recibió el mensaje. Estas pruebas infalsificables de no repudio pueden ser verificadas por una tercera entidad confiable por los dos.

La forma de conseguir la propiedad de no repudio es aplicando tipos específicos de firma digital en el intercambio de la información [6]. La firma digital es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente determinar la entidad originadora de dicho mensaje, al mismo tiempo que permite evaluar si el mensaje ha sido alterado desde que fue firmado por el emisor. La firma digital se aplica en aquellas áreas donde es importante verificar la autenticidad y la integridad de ciertos datos, ya que proporciona una herramienta para detectar la falsificación y la manipulación del contenido.

#### *2.1.4.2 Autenticación*

Es la propiedad que controla el acceso de los usuarios a un sistema y del mismo modo también controla los recursos e información a los que un usuario tiene acceso [7]. En un sistema informático esto se consigue mediante un control de acceso con usuarios y contraseñas.

La autenticación de un usuario involucra 3 actores:

- **Solicitante**: es el propio usuario, el cual proporciona la información necesaria para demostrar que es quien dice ser. Esta información se emplea a modo de método de identificación y, por ejemplo, puede ser un nombre o un número identificativo.

- **Autenticador:** es la entidad encargada de verificar que la información proporcionada por el solicitante es correcta. Dicho de otro modo, el autenticador está asociado al método o conjunto de métodos mediante los cuales el usuario prueba que es quien dice ser. Un ejemplo de este tipo de métodos de verificación es una contraseña.

- **Autoridad de seguridad:** está encargada de custodiar y verificar las credenciales de autenticación. Esto es, la autoridad de seguridad se encarga de controlar el acceso del solicitante a los recursos mediante la validación de la información aportada tanto por el identificador como por el autenticador.

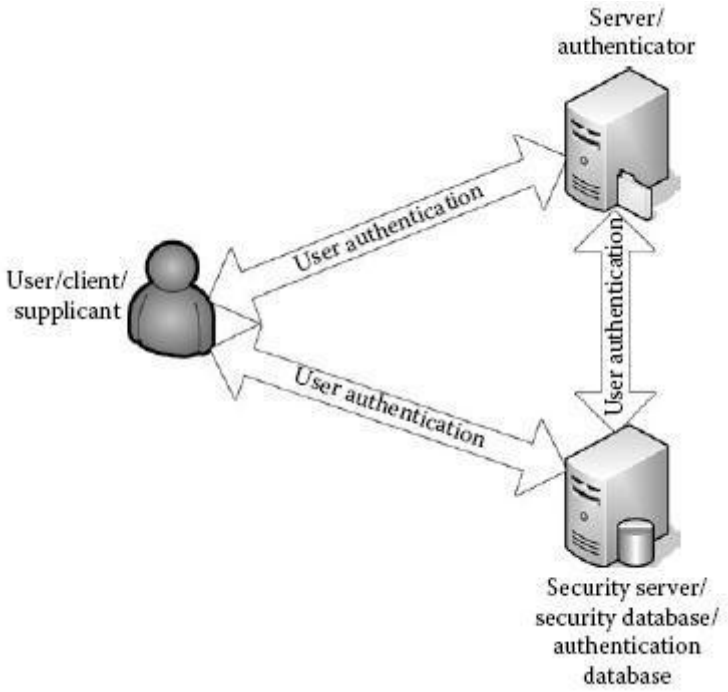


*Ilustración 3: Actores involucrados en la autenticación*

Por otro lado, el uso adecuado de un método de identificación por parte del autenticador requiere que se cumplan el siguiente conjunto de requerimientos:

- Cada valor debe ser único para cada cuenta de usuario
- Se debe seguir un estándar en los valores posibles
- Los valores no deben ser descriptivos de la función que desempeña el usuario
- Los valores no se deben compartir entre los usuarios

Finalmente, cabe distinguir tres tipos de autenticación según el procedimiento que aplique la autoridad de seguridad:

- Tipo 1: *algo que **se sabe***. En este caso la autoridad valida el ingreso de un usuario si este proporciona información que solo él conoce (e.g., una contraseña o un PIN). Este tipo de autenticación es la menos segura y menos cara.

- Tipo 2: *algo que **se tiene.*** Ahora el usuario debe aportar un token identificativo (una llave, una tarjeta, etc.) si quiere que la autoridad le dé acceso a cierto recurso o conjunto de recursos. Este mecanismo de autenticación es más seguro y caro que la anterior: Ejemplo: llave, una tarjeta de acceso.

- Tipo 3: *algo que **se es***. El último procedimiento de autenticación consiste en constatar que el solicitante posee una propiedad física unívocamente asociada a él. De modo típico este control se identifica con técnicas biométricas como el escáner de retina, huella dactilar, etc… Esta técnica es la más segura y cara, y además no goza de una buena aceptación por parte de los usuarios.

Para conseguir una **autenticación** más segura, es recomendable implementar dos o tres formas de las anteriormente descritas.

## 2.2 Aspectos de los sistemas de autenticación

En los sistemas de autenticación hay dos factores igual de importantes: la usabilidad y la seguridad [8]. El sistema de información más seguro del mundo, si no es fácil de usar por los usuarios, nadie lo usará.

En la mayoría de los casos, cuanto más fácil de usar es un sistema de seguridad menos seguro se vuelve, ya que en el diseño para buscar esa usabilidad se pierden mecanismos que aumentan la seguridad. Pero ¿hasta qué punto se puede reducir la seguridad de un sistema para aumentar la usabilidad? la respuesta depende de la sensibilidad de la información que se vaya a guardar en ese sistema. Al final, se trata de una cuestión de economía, ¿vale la información que se puede robar el esfuerzo y recursos necesarios para romper los mecanismos que la protegen? Si la respuesta a esta pregunta es positiva, se debe implementar un sistema más seguro y difícil de romper aunque con ello se reduzca la usabilidad.

Por poner un ejemplo sobre la seguridad y las contraseñas de los usuarios. Las contraseñas que se usan para acceder a cuentas de correo, redes sociales, foros o páginas web donde se haya registrado un usuario, deben de ser fáciles de recordar pero a la vez difíciles de descifrar. La gran mayoría de usuarios siempre usan el mismo usuario y contraseña para acceder a estos sitios, de esta forma les es más fácil acordarse y es mucho más usable, pero esto resulta muy inseguro, ya que con averiguar la contraseña y usuario de una cuenta, podrían acceder a todas la cuentas de ese usuario en otros servicios. Y además estas contraseñas suelen ser nombres o cumpleaños de personas conocidas o mascotas para que sean más fáciles de recordar. En otros casos hay usuarios que tienen un usuario y contraseña distinto para cada servicio en que este registrado, lo cual seguramente provocará que se equivoque más de una vez al intentar acceder a estos. Este caso es mucho más seguro pero menos usable [9].

Como hemos comentado el acceso a la mayoría de cuentas en los distintos servicios se realiza mediante un usuario y una contraseña (numérica o alfanumérica), esto provoca que los usuarios reutilicen usuarios y contraseñas en las distintas cuentas que puedan tener. Si se cambiase la forma de autenticarse en estos sistemas, se podría aumentar considerablemente la seguridad, manteniendo la usabilidad. En este contexto y como objetivo de este proyecto,

las contraseñas visuales ofrecen la posibilidad de mejorar la seguridad manteniendo la usabilidad.

### 2.2.1 Seguridad

Son muchos los aspectos que hay que tener en consideración en la seguridad a la hora de diseñar cualquier sistema de autenticación, que van desde la codificación y almacenamiento de las contraseñas hasta la capacidad para resistir ataques tanto en local como desde la red [10]. A continuación comentamos varios de esos aspectos:

- **Almacenamiento de contraseña:** en el diseño de un sistema de autenticación es importante la forma en la que se vaya a almacenar los usuarios y contraseñas de todas las personas registradas en ese sistema. Estos datos, sobretodo la contraseña, se deben almacenar de manera cifrada. La forma más segura es almacenando el resultado de aplicar una función hash a la contraseña y al usuario. En un sistema online, cuando se envía la solicitud de ingresar en el sistema se debe enviar ya información cifrada y compararla en el destino. Si se cifra y compara en el destino, una tercera persona podría estar escuchando ese canal de comunicación y capturar la información que deseamos ocultar. En un sistema local no existe ese problema al no haber una comunicación en la red, aun así la base de datos donde se almacena todo debe estar cifrada para evitar ataques contra ella.

- **Espacio de claves:** el espacio teórico claves de una contraseña es el número total de contraseñas posibles. Este espacio teórico asume una distribución equiprobable de las contraseñas. El tamaño de este conjunto depende del número total de "caracteres posibles" y de la longitud de la contraseña. En las contraseñas visuales, los caracteres posibles son los puntos donde se puede pulsar en la imagen. Cuanto más grande es el espacio de claves, más seguro es el sistema ya que será más resistente a cierto tipo de ataques como el de fuerza bruta. En los sistemas de autenticación con una contraseña numérica, tiene un espacio de claves muy pequeño en comparación con otros sistemas con contraseñas alfanuméricas o visuales. En sistemas de autenticación con contraseñas alfanumérica el espacio de claves varía mucho ya que depende de los caracteres permitidos, solo letras o también caracteres especiales como signos de exclamación o comillas. En los sistemas con contraseñas visuales el espacio de claves depende íntegramente de las zonas posibles donde se pueda pulsar, en un punto más adelante comentaremos ciertos tipos de contraseñas visuales y las zonas posibles donde se puede pulsar.

- **Resistencia a ataques:** para que una contraseña se considere segura debe ser larga y no debe guardar relación con el usuario, es decir, no deben ser nombres de familiares, mascotas, cumpleaños, aniversarios… las contraseñas para ser seguras deben ser aleatorias. Si una contraseña es larga, la hace más resistente a ataques como los de fuerza bruta o *shoulder surfing*. Si una contraseña es aleatoria la hace más resistente a ataques como los de diccionario o la ingeniería social.

### 2.2.2 Usabilidad

La definición de usabilidad que tomamos es la que aporta la ISO (ISO 924111) [11], donde la usabilidad es interpretada como la unión tres factores:

- **Efectividad**: es la precisión y grado de éxito en las tareas que realiza un usuario, como puede ser en un sistema de autenticación.

- **Eficiencia**: es la relación entre el grado de éxito en las tareas que realiza un usuario y los recursos utilizados para conseguirlo.

- **Satisfacción**: es la evaluación positiva del usuario hacia el sistema.

El que un sistema de autenticación se utilice o no en muchos casos depende de su usabilidad más que de la seguridad que aporte. Por esto vamos describir la forma de cuantificar la usabilidad de un sistema sin tener en cuenta la seguridad que puedan aportar.

La usabilidad se puede medir de forma cualitativa por la visión que ofrezca un usuario sobre el sistema. O se puede medir de una forma cuantitativa viendo la eficiencia, efectividad o la facilidad para la memorización de las contraseñas para acceder:

- **Eficiencia:** podemos medir la eficiencia de un sistema de autenticación por el tiempo que lleva a un usuario completar un acceso con o sin éxito. En otras palabras, un sistema en cual necesitemos mucho tiempo completar el acceso será menos usable que otro con el que podamos acceder más rápido.

- **Efectividad:** la efectividad la podemos medir por el número de intentos que necesita un usuario de media para poder acceder con éxito al sistema. Si es muy complicado acceder y la mayoría de los intentos son erróneos puede llevar a la frustración del usuario, algo que en un sistema sencillo y de fácil acceso no pasaría.

- **Memorización:** la capacidad de memorización de las contraseñas para acceder a un sistema es un punto importante, la elección de las contraseñas alfanuméricas en la gran mayoría de usuarios está basada en nombres de amigos, familiares, mascotas o fechas importantes. Estudios demuestran que las personas tienen más facilidad a la hora de memorizar una imagen que una palabra, por ello las contraseñas visuales ofrecen una mayor usabilidad que una contraseña alfanumérica aleatoria.

### 2.2.3 Problemas de la autenticación

La mayoría de sistemas únicamente implementan la forma más sencilla de autenticación. Además, las contraseñas en estos sistemas no suelen cambiar, al no ser que la contraseña expire o el usuario se olvide de ella. Por esta razón estos sistemas son más vulnerables a los ataques de reinyección, en la cual un atacante puede adquirir la contraseña de un usuario mediante el espionaje del canal de comunicación entre el usuario y el sistema en una red. Otra forma de adquirir la contraseña de un usuario es mediante el uso de *Keyloggers*, un programa capaz de almacenar las entradas en el teclado de un usuario, estos programas se instalan sin el conocimiento del usuario con el fin de espiarle. En los casos en los que un tercero ha conseguido la contraseña de un usuario de forma ilícita y para evitar que pueda ingresar en

ese sistema, algunos implementan varios tipos de autenticación simultáneas o **sistemas de autenticación multicanal**.

Los sistemas de autenticación multicanal son sistemas que implementan dos o más sistemas para que un usuario pueda ingresar de forma correcta, como por ejemplo una contraseña y una tarjeta. Esto aumenta de forma considerable la seguridad de cualquier sistema ya que aunque una tercera persona consiguiera averiguar la contraseña, no podría acceder sin tener la tarjeta.

Otros sistemas de autenticación multicanal usan contraseñas de un solo uso, estas contraseñas las envían al correo o mediante un mensaje al móvil una vez el usuario ha intentado ingresar por el primer método implementado. De este modo, estas contraseñas se usan como segundo método de autenticación. Los bancos usan mucho este sistema en las compras por internet, donde una vez introducidos los datos de la tarjeta se envía al móvil un código para validar la transacción online.

#### *2.2.3.1 One-Time Passwords*

Las contraseñas que se usan para acceder a muchos sistemas no suelen cambiar mucho, de forma que son vulnerables a ataques de Replay o reinyección. Las *One-Time Passwords* u OTP surgen para paliar esa vulnerabilidad. Son contraseñas de un solo uso, de forma que cada vez es distinta. Esto soluciona el problema de los ataques de reinyección.

En la generación de estas contraseñas la aleatoriedad juega un papel muy importante, de otra forma sería fácil deducir una OTP futura a partir de las anteriores. Existen varios métodos de generación de OTP.

- **Sincronizadas con la hora:** con este método se usa la hora actual del sistema como fuente de entropía en la generación de las OTP. El problema de este método es la sincronización de varios sistemas, ya que cada uno tiene su hora. Y a la hora de generar las OTP, cada sistema creara una distinta y por lo tanto no podrán ser validadas entre ellos.

- **Basadas en OTP anteriores:** con este método se usa una función para generar las OTP a partir de la OTP anterior. Primero se genera un número finito de OTP y se usan en orden inverso al orden en que se generaron. Para que esta práctica sea segura, la función utilizada no debe tener inversa ya que en ese caso una persona que sepa varias OTP seguidas, podría calcular esa inversa y averiguar la siguiente OTP valida.

- **Otras fuentes de entropía:** ciertos sistemas, como los dispositivos móviles actuales, tienen multitud de sensores, como de movimiento, gravedad, humedad, GPS, pulsaciones en la pantalla, proximidad… que se pueden utilizar como *fuente de entropía* en la función que genera las OTP.

En todos los casos, la función que genera las OTP son *funciones hash* debido a que es muy complejo y costos el calcular la función inversa. Aunque las OTP nos ofrecen un gran aumento en la seguridad, hay que destacar sus limitaciones:

- Las OTP no ofrecen protección frente a la intercepción de paquetes en la red.

- Las OTP no protegen la privacidad de la información.

- Son vulnerables a un ataque por ingeniería social, si un atacante consigue que un usuario le diga las últimas OTP utilizadas podría determinar las siguientes. Si estas se generan a partir de las anteriores.

- Las OTP dependen totalmente de dificultad por encontrar una función inversa de la función que las genera. Actualmente ya hay funciones hash que han sido comprometidas y no se recomienda su uso, entre ellas están las funciones MD4, MD5 o SHA-1 [12].

## 2.3 Contraseñas Visuales

Como hemos comentado anteriormente, las contraseñas visuales ofrecen un cambio en el paradigma del control de acceso mediante un usuario y una contraseña alfanumérica que se implementa en la gran mayoría de sistemas. Las contraseñas visuales fueron descritas por primera vez en 1996 por Greg E. Blonder [13]. En esta descripción, una imagen aparece en la pantalla y el usuario debe pulsar sobre algunas regiones concretas de la imagen. Si las regiones pulsadas son correctas, el usuario se habría autenticado de manera satisfactoria [14].

### 2.3.1 Tipos de autenticación usando Contraseñas Visuales

Las contraseñas visuales se pueden clasificar principalmente en cuatro grandes grupos [15]:

- **Reconocimiento:** el usuario debe encontrar y reconocer algo sobre una o varias imágenes para poder autenticarse.

- **Memorización de patrones:** el usuario debe reproducir un patrón o dibujo que haya creado como su contraseña para poder autenticarse.

- **Memorización de puntos claves:** el usuario debe seleccionar una o varias partes concretas de una imagen para poder autenticarse.

- **Esquemas híbridos: s**on sistemas que combinan varios sistemas anteriormente nombrados.

A continuación vamos a describir varios tipos de autenticación usando contraseñas visuales. En todas ellas para poder autenticarse con éxito hay que pulsar en los puntos correctos para poder autenticarse. Todos estos métodos permiten modificaciones para aumentar la seguridad o la usabilidad.

- **Pass-Go:** este sistema está basado en la memorización de una serie de puntos dentro de una cuadricula. El usuario puede seleccionar tanto aristas como intersecciones para formar un patrón. En este sistema no importa el orden de entrada sino la forma final, de modo que coincida con el patrón que el usuario definió para autenticarse.

- **UYI** (*Use Your Illusion*): este sistema está basado en el reconocimiento de imágenes. El usuario elige una imagen o una serie de imágenes clave, el sistema las irá mostrando de forma distorsionada junto a otras imágenes señuelo. El usuario para autenticarse tiene que ser capaz de reconocer las imágenes que ha elegido estando estas distorsionadas.

- **TAPI** (*Touchsreen Auth. Using Partitioned Images*): este sistema también está basado en el reconocimiento y memorización de claves en un conjunto de imágenes. Usa un conjunto de imágenes, colocadas de forma aleatoria, divididas en varias partes. El usuario para poder autenticarse debe pulsar en las zonas correctas de cada imagen y en el orden correcto.

- **CCP** (*Cued Click Points*): este sistema se basa en la memorización de una serie de puntos dentro de una imagen. Puesto que pulsar dos veces sobre el mismo pixel es muy difícil y teniendo en cuenta que cuando pulsamos la pantalla pulsamos más de un pixel a la vez, este sistema divide la imagen en una cuadricula invisible para el usuario. Para no perder seguridad, dando falsos positivos, las cuadriculas deben ser lo suficientemente pequeñas para que al pulsar en un sitio distinto no lo reconozca como correcto, pero lo suficientemente grandes para no pulsar dos zonas a la vez.

### 2.3.2 Contraseñas Visuales vs Contraseñas Alfanuméricas

Para poder medir realmente cual es de estos métodos es más seguro, analizaremos distintos ataques y las ventajas que ofrece un sistema sobre el otro para llegar a una conclusión:

- **Ataques de Fuerza Bruta:** la principal defensa frente a un ataque de fuerza bruta, es tener un espacio de claves lo suficientemente grande para que este tipo de ataques sean muy costos y tarden en ejecutarse. Una contraseña alfanumérica, sin caracteres especiales, tiene una espacio de claves de alrededor de $35\text{^}n$ posibilidades, donde *n* es el número de caracteres. Algunos tipos de contraseña visual como TAPI o CCP pueden superar ampliamente el espacio de claves anterior, ya que el modo TAPI tiene un espacio de claves de $q^n \times \frac{m!}{(m-p)!}$ Donde *q* es el número de zonas sensibles posibles, *n* es el número de pulsaciones, *p* el número de imágenes distintas utilizadas y *m* el número total de imágenes, mientras que el modo CCP dependerá del número de regiones en las que se haya dividió la imagen, además el modo TAPI cuenta con la posición aleatoria de las imágenes, por lo que un ataque de fuerza bruta involucraría un mayor coste computacional. Por estas razones creemos que las contraseñas visuales son más seguras que las contraseñas alfanuméricas frente a los ataques de fuerza bruta.

- **Ataques de Diccionario:** los ataques de diccionario consisten en intentar inferir el valor de una contraseña probando todas las palabras del diccionario. En una contraseña visual no hay un diccionario que contenga palabras formadas por entradas del ratón, por esta razón una contraseña visual no puede ser o es muy difícil que sea atacada por

este método. De modo que la contraseña visual ofrece una ventaja frente a estos ataques.

- ***Spyware*:** salvo por algunas excepciones, los *Keylogger* no pueden ser usados para romper las contraseñas visuales. Los *spywares* de seguimiento del ratón no son del todo efectivos, ya que no basta con saber la posición del puntero. Para que este tipo de software fuera efectivo tendría que saber en tiempo real lo que se muestra en la pantalla, el tamaño de esta y la posición del ratón. Así pues, frente a este tipo de ataques, las contraseñas visuales ofrecen una mayor seguridad.

- ***Shoulder Surfing*:** tanto las contraseñas alfanuméricas como las visuales son vulnerables a estos ataques. Únicamente las técnicas basas en reconocimientos faciales, voz o similares son capaces de resistir estos ataques.

- **Ingeniería Social:** las contraseñas alfanuméricas, en la gran mayoría de los casos son nombres de personas conocidas, mascotas o fechas importantes como cumpleaños o aniversarios. De forma que con una mínima investigación sobre un usuario, se puede restringir bastante el espacio de claves. Sin embargo una contraseña visual, como pueda ser el sistema TAPI o CCP, es muchísimo más difícil de averiguar los posibles lugares donde el usuario pulsaría, ya que requería una investigación más exhaustiva. Frente a otros tipos de ataques, incluidos en la ingeniería social, las contraseñas visuales también ofrecen una ventaja sobre las alfanuméricas, ya que es mucho más difícil intentar decir a otra persona cuales son los puntos exactos que forman la contraseña que decir la contraseña alfanumérica.

Como conclusión de este análisis preliminar podemos afirmar que las contraseñas visuales parecen proporcionar una mayor seguridad que las contraseñas alfanuméricas o numéricas tradicionales frente a los distintos tipos de ataques que hemos comparado [16],[17].

### 2.3.3 Problemas de las contraseñas visuales

Ya nombradas y explicadas las ventajas que ofrecen las contraseñas visuales respecto a las contraseñas alfanuméricas, a continuación explicamos varios problemas que existen a la hora de desarrollar un sistema de autenticación basado en contraseñas visuales.

#### *2.3.3.1 Discretización*

La *discretización* es una técnica usada en las pulsaciones en las contraseñas visuales para tolerar la varianza de la entrada mediante una aproximación de acuerdo con el pixel de la pantalla más próximo. Cuando pulsamos en un imagen siempre pulsamos un conjunto de pixeles, por ello es necesario determinar la zona a la que pertenece esa pulsación. Para ello se lleva a cabo la división de la pantalla mediante unas cuadriculas, de forma que cada celda constituye la zona sensible mediante la cual se codificará la entrada proporcionada por el usuario al pulsar la pantalla [18].

En el sistema CCP que hemos definido la imagen se divide en una cuadricula, si los cuadrados que conforman la cuadricula es demasiado grande, el porcentaje de falsos

positivos[1] a la hora de autenticarse aumenta de forma considerable ya que disminuye el espacio teórico de claves y puntos lejanos en la imagen pueden formar parte de un mismo cuadrado. Por el contrario si los cuadrados son muy pequeños, cuando pulsemos para introducir la contraseña es posible que se pulsen varios cuadrados a la vez, dando como resultado un aumento en los falsos negativos[2]. Con los demás sistemas de autenticación que hemos explicado anteriormente ocurre más o menos lo mismo, si las zonas son muy pequeñas es muy difícil autenticarse con éxito.

Esto puede transformarse en un problema mayor, ya que no todos los dispositivos tienen el mismo tamaño ni la misma resolución. Por esto las contraseñas visuales son poco usables en dispositivos con pantalla pequeña, ya que las zonas sensibles para generar las contraseñas serían demasiado pequeñas para poder pulsarlas y autenticarse con éxito.

A continuación ilustraremos con un ejemplo visual lo anteriormente explicado:

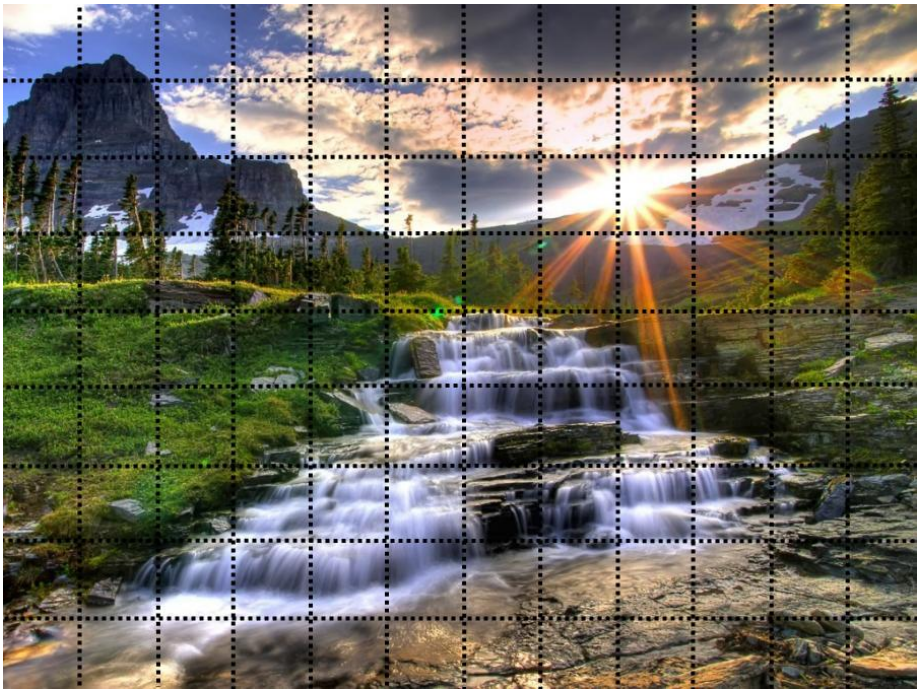

*Ilustración 4: Imagen con una cuadricula de 11x12*

[1] Los falsos positivos son autenticaciones positivas cuando la entrada es errónea.
[2] Los falsos negativos son autenticaciones negativas cuando la entrada en correcta.

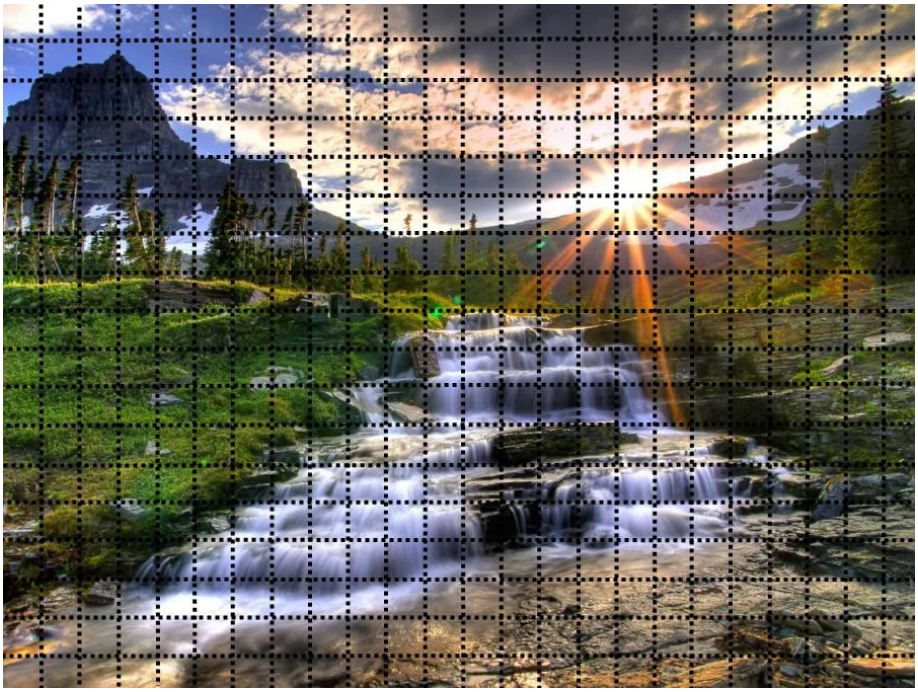

*Ilustración 5: Imagen con una cuadricula de 22x24*

Como puede verse al disminuir el tamaño de la cuadricula, se gana en espacio de claves pero se puede perder precisión al intentar pulsar con el dedo en una pantalla táctil ya que cuanto más pequeños sean los cuadros más fácil es pulsar varios a la vez y esto dificulta enormemente el intentar acceder al sistema.

#### *2.3.3.2 Puntos de interés en una imagen*

Muchas aplicaciones, como las que generan fotos panorámicas, necesitan relacionar dos o más imágenes viendo los puntos que tienen en común para poder unirlas en una sola imagen. El método de comparar pixel por pixel entre todas las imágenes para poder unirlas es demasiado costoso computacionalmente. Sin embargo un usuario puede relacionar fácilmente dos imágenes viendo las coincidencias de ciertos puntos que de alguna forma resultan interesantes. Tales puntos se denominan **puntos de interés**.

Existen distintos detectores de puntos de interés y cada uno está programado para considera diferentes puntos o zonas de interés para una misma imagen [19]–[22]. Algunos buscan puntos de alta simetría, otros encuentran áreas donde existe una gran variación en la textura y otros sitúan los puntos de interés en las esquinas de los objetos.

El uso de los puntos de interés para encontrar la correspondencia de puntos entre varias imágenes es un paso clave en muchas técnicas de procesamiento de imágenes y visión artificial. Algunos casos de uso son:

- Unión de imágenes para crear panorámicas
- Detección y reconocimiento de objetos
- Seguimiento del movimiento
- Visión de la Inteligencia Artificial (I.A.)

Volviendo con las contraseñas visuales, la detección de puntos de interés en una imagen puede reducir considerablemente el espacio de entradas con el que intentar realizar un ataque. Con las siguientes imágenes explicaremos este fenómeno:

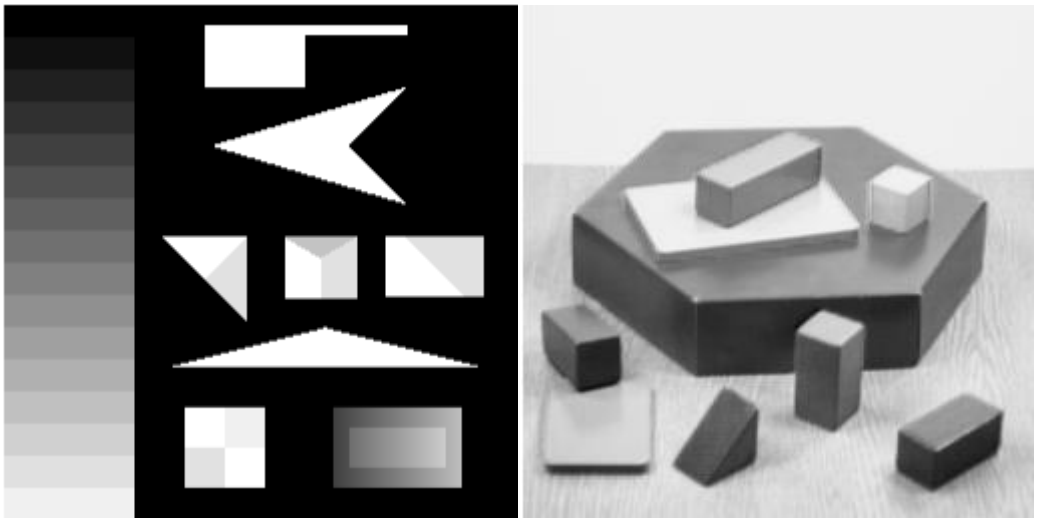

*Ilustración 6: Imágenes antes de la detección de puntos de interés*

Si estas imágenes formasen parte de un sistema de autenticación de contraseña visual, tras aplicarle los algoritmos de detección de puntos de interés podríamos verlas de esta forma:

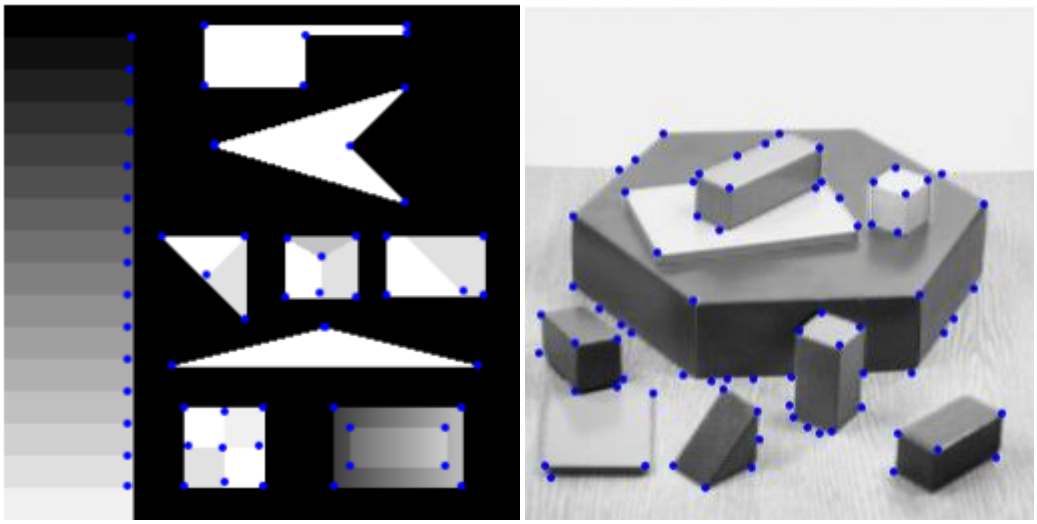

*Ilustración 7: Imágenes después de la detección de puntos de interés*

Aunque no cubre todos los posibles puntos, si abarcan un gran porcentaje de puntos que la mayoría de usuarios elegirían como parte de sus contraseñas. Dicho de otra forma, el análisis de puntos de interés permite deducir las posibles áreas que un usuario puede elegir para guiar su autenticación. Tal detección implica una reducción del espacio de búsqueda de un posible atacante, lo que en último término significa una reducción considerable del coste computacional de un ataque por fuerza bruta.

### 2.3.4 Aplicación de las contraseñas visuales

Las contraseñas visuales ya forman parte de muchos procesos de autenticación que usamos diariamente. Por ejemplo en la banca por internet, para poder iniciar sesión debes introducir la contraseña pulsando en la imagen de un teclado que va cambiando de posición en la pantalla según se va introduciendo la contraseña:


*Ilustración 8: Ejemplo de contraseña visual en el acceso al banco*

Aunque la contraseña sea alfanumérica, podemos considerarla como visual ya que es necesario pulsar sobre una imagen y si por alguna razón el usuario tiene en su ordenador un *Keylogger*, con este sistema se evita que capture la contraseña ya que no se introduce mediante el teclado [23], [24].

Otros sistemas que utilizan con mucha frecuencia las contraseñas visuales son los dispositivos móviles. Ya que muchos usuarios eligen de dibujar un patrón en la pantalla para desbloquear el dispositivo en vez de introducir un código, mientras que otros casos basta con desplazar una imagen de la pantalla hasta una zona determinada para desbloquearlos:

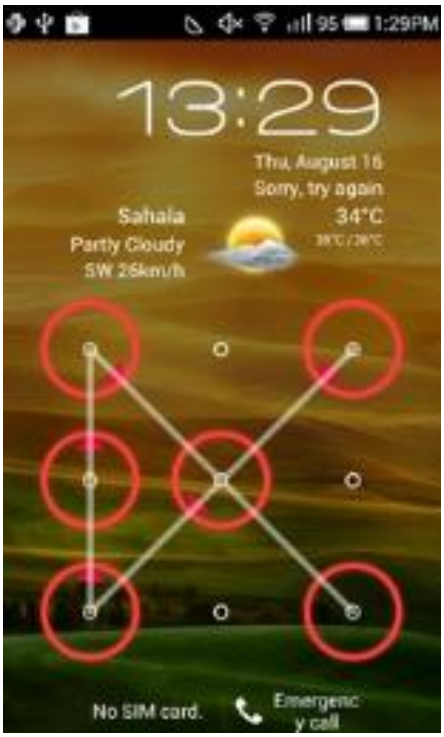


*Ilustración 9: Ejemplo de contraseña visual para desbloquear un móvil*

El ámbito definido por el uso de contraseñas visuales en dispositivos móviles constituye el foco de este trabajo. En efecto, nuestro interés es estudiar las virtudes y limitaciones de estos procedimientos de validación de usuario en una de las tecnologías más importantes en la actualidad. A este respecto, además, nos interesa abordar la familia de dispositivos móviles que goza de mayor aceptabilidad por parte del público. Por este motivo, en los siguientes capítulos desarrollaremos la problemática de las contraseñas visuales articulando nuestra exposición alrededor de su implantación en móviles con sistema operativo Android.

# 3. Android

Para conseguir una alta usabilidad hay que usar tecnologías que sean fáciles de usar. La tecnología que vamos a usar en este sentido son los dispositivos móviles Android.

Android se ha convertido en el sistema operativo para dispositivos móviles más usado en todo el mundo. En el este capítulo veremos un breve resumen de su historia y evolución, además explicaremos sus principales características.

## 3.1 Historia

Android fue inicialmente desarrollado por Android Inc. una pequeña compañía comprada por Google en 2005. A partir de este momento, Google en compañía de la *Open Handset Alliace* se encargaron de su desarrollo. Android es un sistema basado en Linux diseñado para dispositivos móviles con pantalla táctil. Una de sus mayores ventajas frente a otros sistemas como *iOS* es que es de código abierto y de esta forma cualquiera puede crear sus propias aplicaciones y *ROMs*.

El primer teléfono en implementar este sistema fue el HTC Dream, que fue lanzado al mercado el 22 de octubre de 2008. Esta primera versión de Android, Android 1.0, incluyó desde el principio soporte para *wifi* y *bluetooth*, también incluyó una aplicación capaz de acceder a los servidores de correo de terceros con soporte para los estándares POP3, IMAP4 y SMTP.14. Tampoco faltaron aplicaciones para dar soporte a los servicios de google más populares como *Google Maps* con *Latitude* y *Street View*, *Google Sync* para sincronizar *Gmail*, Contactos y Calendario, *Google Search*, *Google Talk* y *YouTube*.

Las versiones de Android reciben el nombre de diferentes postres, además cada postre empieza por una letra distinta siguiendo un orden alfabético:

| Versión | Nombre |
|---|---|
| 1.0 | Apple Pie |
| 1.1 | Banana Bread |
| 1.5 | Cupcake |
| 1.6 | Donut |
| 2.0/2.1 | Froyo |
| 2.3 | Gingerbread |
| 3.* | Honeycomb |
| 4.0 | Ice Cream Sandwich |
| 4.1/4.2/4.3 | Jelly Bean |
| 4.4 | KitKat |
| 4.6 o 5.0 | Lime Pie |

*Tabla 1: Versiones de Android*

Desde que lanzaron la primera versión, Android ha ido poco dominando el sector de los dispositivos móviles. Actualmente Android tiene una cuota de mercado superior al 85% en España, pero es igualmente alta en otros países como Alemania con un 75.7% o China con 80.3%. En otros países como Estados Unidos la diferencia entre Android, situado a la cabeza con un 55% del mercado, y el segundo puesto es menor, iOS con un 38.7%.

| Germany | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 71.9 | 75.7 | 3.8 |
| BlackBerry | 0.6 | 0.7 | 0.1 |
| iOS | 18.8 | 15.3 | -3.5 |
| Windows | 5.4 | 7.5 | 2.1 |
| Other | 3.2 | 0.7 | -2.5 |

| USA | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 51.1 | 55.0 | 3.9 |
| BlackBerry | 0.7 | 0.6 | -0.1 |
| iOS | 43.6 | 38.7 | -4.9 |
| Windows | 4.1 | 5.3 | 1.2 |
| Other | 0.5 | 0.4 | -0.1 |

| GB | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 58.3 | 54.0 | -4.3 |
| BlackBerry | 5.1 | 3.4 | -1.7 |
| iOS | 29.0 | 32.1 | 3.1 |
| Windows | 6.7 | 10.1 | 3.4 |
| Other | 0.9 | 0.3 | -0.6 |

| China | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 71.0 | 80.3 | 9.3 |
| BlackBerry | 0.2 | 0.1 | -0.1 |
| iOS | 24.4 | 17.9 | -6.5 |
| Windows | 1.3 | 1.0 | -0.3 |
| Other | 3.1 | 0.7 | -2.4 |

| France | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 61.9 | 67.7 | 5.8 |
| BlackBerry | 5.1 | 1.4 | -3.7 |
| iOS | 23.1 | 21.1 | -2.0 |
| Windows | 6.3 | 8.3 | 2.0 |
| Other | 3.7 | 1.5 | -2.2 |

| Australia | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 58.8 | 58.5 | -0.3 |
| BlackBerry | 0.4 | 0.7 | 0.3 |
| iOS | 34.8 | 33.9 | -0.9 |
| Windows | 3.5 | 5.0 | 1.5 |
| Other | 2.5 | 1.8 | -0.7 |

| Italy | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 58.5 | 67.6 | 9.1 |
| BlackBerry | 1.9 | 1.4 | -0.5 |
| iOS | 22.6 | 13.6 | -9.0 |
| Windows | 11.9 | 16.1 | 4.2 |
| Other | 5.0 | 1.3 | -3.7 |

| Japan | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 43.7 | 44.8 | 1.1 |
| BlackBerry | 0.0 | 0.0 | 0.0 |
| iOS | 53.0 | 54.9 | 1.9 |
| Windows | 0.4 | 0.2 | -0.2 |
| Other | 2.9 | 0.1 | -2.8 |

| Spain | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 92.0 | 88.0 | -4.0 |
| BlackBerry | 1.0 | 0.0 | -1.0 |
| iOS | 4.5 | 6.3 | 1.8 |
| Windows | 0.8 | 4.7 | 3.9 |
| Other | 1.6 | 0.9 | -0.7 |

| EU5 | 3 m/e Feb 2013 | 3 m/e Feb 2014 | % pt. Change |
|---|---|---|---|
| Android | 66.8 | 68.9 | 2.1 |
| BlackBerry | 2.9 | 1.6 | -1.3 |
| iOS | 21.1 | 19.0 | -2.1 |
| Windows | 6.3 | 9.7 | 3.4 |
| Other | 3.0 | 0.9 | -2.0 |

*Ilustración 10: Comparativo de las cuotas de mercado*

## 3.2 Características

Como hemos dicho anteriormente, Android es un sistema operativo, además de una plataforma software, basada en un núcleo Linux. Diseñada en un principio para dispositivos móviles, Android permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptadas por Google mediante el lenguaje de programación Java.

Android es una plataforma de código abierto, esto quiere decir que cualquier desarrollador puede crear y desarrollar aplicaciones escritas con cualquier lenguaje y compilarlas a código nativo de ARM (API de Android). Sin embargo, es Google quien ha publicado la mayoría del código fuete de Android bajo la licencia de Software Apache, una licencia de software libre y de código abierto a cualquier desarrollador [25].

Entre sus principales características encontramos:

- **Framework de aplicaciones:** permite el reemplazo y la reutilización de los componentes.
- **Navegador integrado:** basado en el motor open Source Webkit.
- **SQlite:** base de datos que se integra directamente con las aplicaciones.
- **Multimedia: s**oporte para medios con formato comunes de audio, video e imágenes.
- **Máquina virtual Dalvik:** base de llamadas de instancias muy similar a Java.
- **Telefonía GSM:** dependiente del terminal.
- **Bluetooth, EDGE, 3g y Wifi:** dependiente del terminal.
- **Cámara, GPS brújula y acelerómetro:** dependiente del terminal.
- **Pantalla Táctil.**

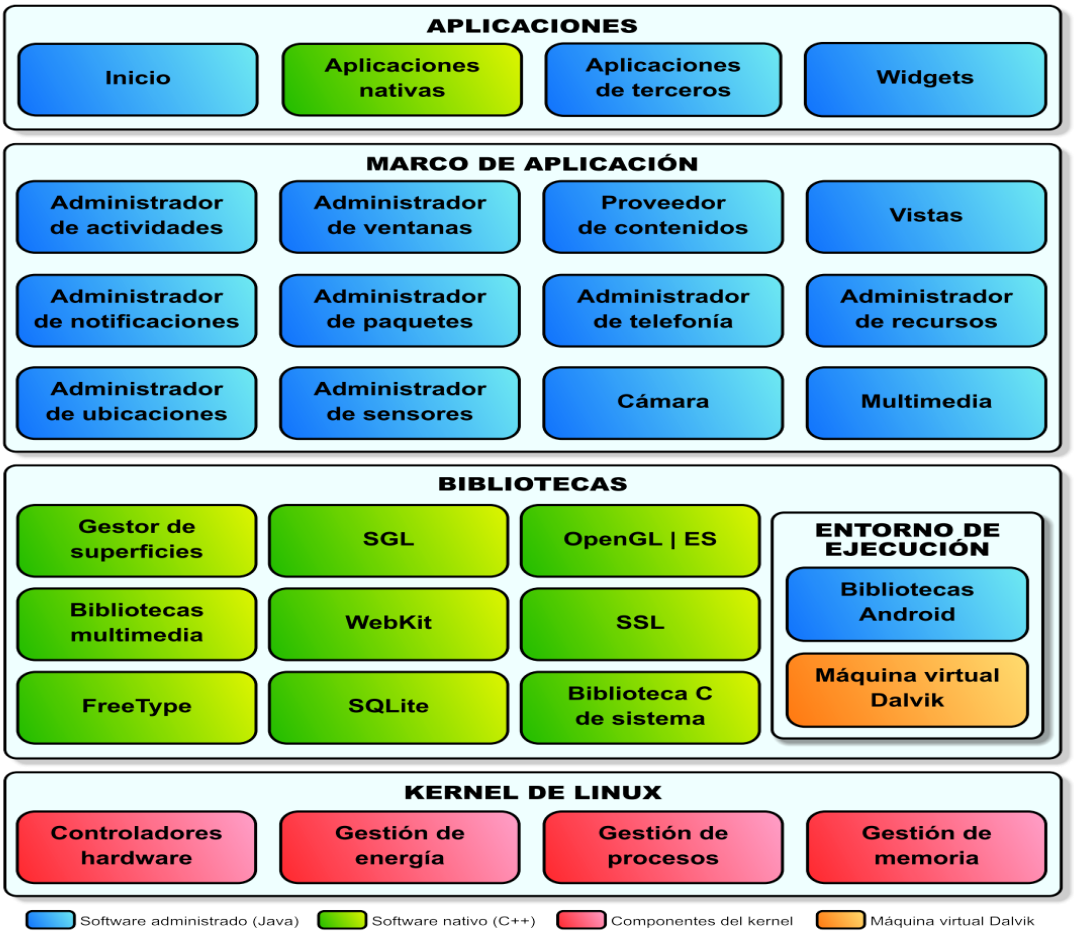


*Ilustración 11: Arquitectura de Android*

### 3.2.1 Arquitectura

Podemos dividir la Arquitectura de Android en varias capas como se aprecia en Ilustración 11: Arquitectura de Android

En la capa superior se encuentran las aplicaciones de usuario tanto las nativas de Android como calendario, mapas, navegador, contactos, como las aplicaciones creadas por los usuarios que se pueden descargar desde la tienda de Android, Google Play.

Todas estas aplicaciones se apoyan en la siguiente capa de la arquitectura, el marco de aplicaciones. De esta forma, todos los desarrolladores tienen acceso a las mismas APIs que usan las aplicaciones base de Android. También está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra puede hacer uso de esas capacidades.

En la siguiente capa se encentran las librerías y el propio *runtime* de Android. Incluye un conjunto de librerías de C/C++ y un set de librerías base que proporcionan la mayor parte de las funciones disponibles. Cada aplicación Android ejecuta su propio proceso, con su propia instancia de la máquina virtual Dalvik, además las aplicaciones se compilan en tiempo de ejecución. El sistema operativo en si está compuesto por unas 12 millones de líneas de código entre las que hay 3 millones de líneas XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

### 3.2.2 Aplicaciones

En los siguientes puntos comentaremos los aspectos más importantes de las aplicaciones Android.

#### *3.2.2.1 Componentes*

Existen una serie de elementos claves que resultan imprescindibles para desarrollar cualquier aplicación en Android. A continuación vamos a dar una breve descripción de estos componentes:

- **Vista (View):** las vistas son los elementos que componen la interfaz de usuario en las aplicaciones, por ejemplo: botones, cuadros de texto, entradas de texto… Todas las vistas derivan de la clase View. Las interfaces se definen usando layout en ficheros XML.

- **Layout:** un Layout es conjunto de vistas agrupadas de una manera determinada. Existen diferentes tipos de layout para organizar las vistas de distinta forma: de forma lineal, en tabla, indicando la posición absoluta de la cada vista… Al igual que las vistas los layout son objetos que descienden de la clase View pero la forma habitual de definirlos es en ficheros utilizando código XML. De esta forma la creación de una interfaz es muy similar a la definición de una página web utilizado código HTML.

- **Actividad (Activity):** las actividades son cada una de las pantallas que forman una aplicación. Su función principal es la creación de la interfaz. Todas las actividades son independientes unas de otras y deben pertenecer a una clase descendiente de Activity.

- **Servicio (Service): s**on componentes que se ejecutan de forma invisible, actualizando datos y las Actividades y disparando Notificaciones. Existen dos tipos de servicios: servicios locales, que son ejecutados en el mismo proceso y servicios remotos, que son ejecutados en procesos separados.

- **Proveedores de Contenidos (Content Providers):** en muchos casos las aplicaciones necesitas compartir información. Android tiene este mecanismo para que puedan compartir datos sin necesidad de comprometer la seguridad de ficheros. Este mecanismo permite acceder a datos de otras aplicaciones, como la lista de contactos, o proporcionar datos a otras aplicaciones.

- **Intención (Intent):** una intención representa la voluntad de realizar alguna acción. Se utiliza cada vez que queramos lanzar una actividad, lanzar un servicio, enviar un anuncio tipo broadcast o comunicarnos con un servicio. Los componentes lanzados pueden ser internos o externos a la aplicación.

- **Receptores de anuncios (Broadcast Receivers):** un receptor de anuncios recibe y reacciona ante los anuncios de tipo broadcast. Estos anuncios pueden ser creados por el sistema o por otras aplicaciones. Los receptores de anuncios no disponen de una interfaz de usuario pero pueden iniciar una actividad.

- **Notificaciones (Notificactions):** mecanismo que permite a las aplicaciones señalar algo a los usuarios sin interrumpir la Actividad en primer plano.

#### *3.2.2.2 Ciclo de vida de las aplicaciones*

Las actividades de una aplicación pasan por una serie de estados desde el momento que se crean hasta su destrucción. Durante este recorrido se invocan automáticamente un conjunto de métodos estándar. La clase Activity proporciona implementaciones por defecto de estos métodos que se pueden sobrecargar. Una actividad puede estar en los siguientes estados:

- **Activa:** cuando la actividad está en primer plano y el usuario puede interactuar con ella.

- **En pausa:** en este estado la actividad sigue visible pero no tiene el foco, por ejemplo cuando un diálogo la tapa parcialmente. La información de la actividad se mantiene pero el sistema puede decidir matar la actividad en ciertas situaciones extremas.

- **Parada:** la actividad está cubierta totalmente por otra actividad. Se mantienen los miembros e información de estado pero es probable que el sistema la mate si necesita memoria.

La transición de los estados y los métodos o funciones que se usan para ello se puede ver en el siguiente diagrama:


*Ilustración 12: Ciclo de vida de una actividad de Android*

### *3.2.2.3 AndroidManifest.xml*

Toda aplicación debe tener un archivo llamado AndroidManifest.xml en la raíz del proyecto. Este fichero contiene la información esencial sobre la aplicación para el sistema Android, esta información la debe tener el sistema antes de que la aplicación pueda ejecutar cualquier código. Esta información es entre otras cosas:

- El nombre del paquete Java. Este nombre sirve como identificador único para la aplicación.
- Contiene la descripción de los componentes de la aplicación, el nombre de las clases que los implementan y sus capacidades.
- Determina qué procesos serán los anfitriones de los componentes de la aplicación.
- Declara los permisos que son requeridos para la ejecución de la aplicación. Como la lectura y escritura en un almacenamiento externo, el acceso a contenido de otras aplicaciones o el acceso a internet.
- Declara también el nivel API mínimo requerido por la aplicación.

Tener la información del Manifest actualizada, correcta y ordenada ayudará mucho a la hora de mantener la aplicación y a su correcto funcionamiento, pero también ayudará a que la aplicación sea visible en el Android Market o Google Play, el cual solo muestra las aplicaciones en las cuales los requisitos de software y hardware se encuentran correctamente especificados en el Manfiest.

El siguiente esquema muestra la estructura general del fichero y todos los elementos que puede contener:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

#### *3.2.2.4 Permisos*

Un punto crítico en la seguridad de los dispositivos móviles, son los permisos que se otorgan a las aplicaciones [26]. En nuestros dispositivos móviles llevamos datos personales que no nos gustaría que una aplicación maliciosa nos robara. Android protege los recursos del hardware y de nuestra información personal mediante los permisos, una importante capa de seguridad en el que toda aplicación que quiera acceder a un recurso debe declararlo en el Manifest.

Cuando vamos a instalar una aplicación que nos bajamos de Google Play, lo primero que nos muestra antes de instalar son los permisos que requiere la aplicación con una lista de los recursos e información a la que necesita acceder para funcionar correctamente.

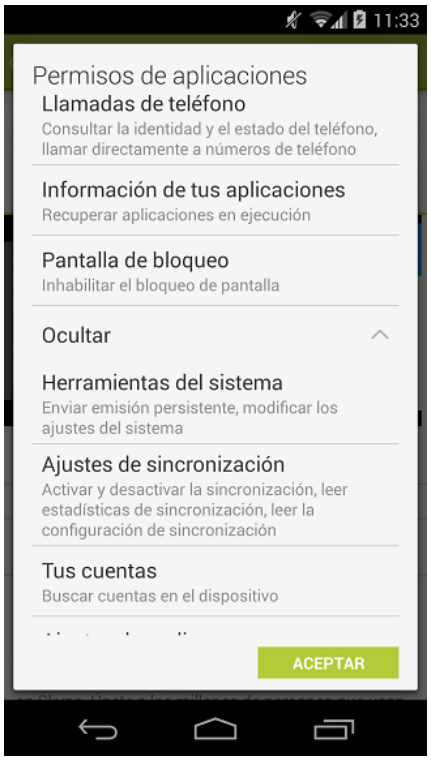


*Ilustración 13: Ejemplo de solicitud de permisos al instalar una aplicación*

El solicitar permisos a los usuarios para acceder a determinadas funciones evita la propagación de aplicaciones maliciosas entre dispositivos Android.

A continuación repasaremos una serie de permisos que existen y con los cuales pueden acceder a nuestros datos personales:

- **Comunicación de red:** estos permisos permiten que una aplicación se pueda conectar a internet para enviar y recibir datos. Si la aplicación cuenta con permisos que accedan a nuestros datos, con estos permisos de red podría enviar nuestros datos a otras personas.

- **Almacenamiento:** estos permisos permiten leer, editar y borrar datos en el almacenamiento USB. De forma que pueden tener acceso a todas las fotos o documentos que almacenemos.

- **Consultar la identidad y el estado del teléfono:** permite que la aplicación acceda a las funciones de teléfono del dispositivo. Permite además acceder a nuestro IMEI, IMSI, al identificador único de 64 bits que Google asigna a cada dispositivo y saber quién nos está llamando.

- **Leer, editar y recibir mensajes de texto:** esta serie de permisos permiten gestionar nuestros SMS/MMS. Son permisos con los que hay que tener mucho cuidado ya que podrían mandar mensajes a servicios de pago sin que el usuario lo sepa.

- **Ubicación:** estos permisos son para que la aplicación pueda saber nuestra ubicación. Aunque este permiso es muy utilizado por buscadores para ofrecernos información según donde nos encontremos, también puede ser utilizado para poder acosar a alguien.

- **Información social y personal:** son permisos que permiten a las aplicaciones acceder a datos muy personales, como nuestros contactos, actividad social, registro de llamadas, calendarios… Hay que tener cuidado que aplicaciones permitimos acceder a nuestros datos.

- **Marcadores e historial:** permiten a una aplicación leer y modificar nuestro historial de navegación y los marcadores. Estos permisos pueden derivar en spam.

- **Buscar y usar las cuentas del dispositivo:** estos permisos hacen que una aplicación pueda ver nuestras cuentas.

# 4. Implementación

En este capítulo explicaremos todos los detalles relativos a la fase de diseño e implementación de este proyecto. Hablaremos sobre el objetivo del proyecto, analizaremos los requisitos funcionales y no funcionales, discutiremos sobre los algoritmos implementados y las decisiones tomadas en su implantación y por último haremos una evaluación de estos algoritmos.

## 4.1 Objetivo

Como hemos comentado al principio de este documento, el objetivo de este proyecto es realizar un estudio sobre los sistemas de autenticación basados en las contraseñas visuales. A lo largo del documento hemos visto las características que deben tener todos los sistemas de autenticación y las ventajas y desventajas que ofrece un sistema de autenticación basado en contraseñas visuales.

El objetivo ahora es comprobar, de forma práctica, si las contraseñas visuales presentan una mejora en la usabilidad y en la seguridad respecto a las contraseñas tradicionales alfanuméricas. Para ello implantaremos varios sistemas de contraseña visuales y uno de contraseña alfanumérica.

## 4.2 Descripción del proyecto software

A continuación vamos a describir las herramientas utilizadas en el desarrollo de este proyecto y haremos una evaluación de los requisitos funcionales y no funcionales que debe cumplir la aplicación desarrollada buscando las características de seguridad y usabilidad que hemos definido anteriormente.

### 4.2.1 Herramientas utilizadas

Para el desarrollo de este proyecto hemos utilizado como entorno de programación Eclipse en su versión Indigo Service Rease 2. Para poder programar en Android se ha usado el IDE proporcionado por Google, Android Developer Tools o ADT, un plugin para Eclipse que ofrece un entorno desarrollo para crear aplicaciones para Android. Otro complemento ofrecido por Google para el desarrollo aplicaciones en Android y que también hemos utilizado es el SDK, Software Development Kit, que incluye un depurador de código, bibliotecas, un simulador de teléfonos basado en QEMU, documentación, ejemplos de código y tutoriales.

Las pruebas unitarias y de integración se ha desarrollado usando Robolectric, un marco de pruebas unitarias creado para probar e impulsar el desarrollo de aplicaciones Android. Las pruebas con usuarios se han realizado directamente con la aplicación, obteniendo un log con las interacciones de los usuarios.

Para la gestión de las referencias bibliográficas para la construcción de este documento se ha utilizado la herramienta Mendeley. Es una herramienta que permite organizar toda la bibliografía del proyecto y permite importar tanto pdfs hasta enlaces web.

### 4.2.2 Requisitos Funcionales

En este punto presentaremos los requisitos funcionales de la aplicación, es decir, aquellos que describen directamente la funcionalidad de la aplicación.

Como hemos comentado en puntos anteriores, los objetivos de todo sistema de seguridad se pueden dividir en dos grupos, seguridad y usabilidad. Por ello dividimos los requisitos funcionales en estos dos grupos.

#### *4.2.2.1 Seguridad*

En este apartado vamos a exponer los requisitos asociados a la seguridad. Puesto que el objetivo de este proyecto es la de realizar un estudio sobre las contraseñas visuales, hemos querido que sea el usuario quien cree su cuenta poniendo los valores que él considere. Pero aun tratándose de un estudio, el almacenamiento y gestión de las contraseñas de los usuarios debe realizarse de forma segura.

De esta forma los requisitos funcionales asociados a la seguridad son:

- **RF1.1:** almacenamiento seguro de claves
    - Usaremos una base cifrada para almacenar las contraseñas. Almacenaremos el resultado hash de las contraseñas en vez de la contraseña en plano o cifrada. La función hash utilizada será SH5, ya que otras funciones como MD4 o MD5 han sido comprometidas y no se recomienda su uso.

- **RF1.2:** repetición de contraseñas al crear un usuario
    - Con el fin de mejorar la seguridad, se pedirá al usuario repetir la contraseña una segunda vez cuando se cree el usuario. Con ello se asegura que el usuario ha introducido correctamente la contraseña con la que quiere registrarse.

- **RF1.3:** métodos de autenticación con un espacio de claves superior a $30\text{^}n$, donde n es el número de “caracteres”.
    - Uno de los aspectos de seguridad para la resistencia de ataques es tener un gran espacio teórico de claves. Por ello los sistemas implementados deben tener un gran espacio de claves.

- **RF1.4:** no debe permitir crear usuarios con el mismo nombre
    - Una de las características de los sistemas de autenticación es que no debe permitir que varios usuarios compartan el mismo nombre.

- **RF1.5:** no debe permitir crear usuario cuando falte algún valor por introducir
    - Si falta algún valor cuando se crea un usuario, ya sea el nombre, la contraseña o la confirmación de la contraseña, no debe permitir crear el usuario.

#### *4.2.2.2 Usabilidad*

En este apartado vamos a exponer los requisitos asociados a la usabilidad del sistema. Dentro del estudio de las contraseñas visuales defendemos su usabilidad debido a que es más fácil recordar una imagen que una palabra. Pero si el sistema no se encuentra correctamente desarrollado y no es intuitivo y usable, no importará si el método de autenticación es mediante contraseñas visuales o alfanuméricas.

De esta forma los requisitos funcionales asociados a la usabilidad son:

- **RF2.1:** interfaz de usuario usable e intuitiva
  - La usabilidad de los sistemas de autenticación no solo se basa en la usabilidad del método de autenticación sino también en la interfaz. Por ello la interfaz debe ser sencilla e intuitiva.

- **RF2.2:** métodos de autenticación sencillos
  - Como hemos comentado anteriormente, es importante la usabilidad del sistema y del método de autenticación. Por esta razón los métodos implementados deben ser sencillos de usar.

- **RF2.3:** métodos de autenticación eficientes
  - Una de las características de la usabilidad es la eficiencia. El tiempo para autenticarse con éxito en la aplicación no debe ser superior al tiempo necesario para la entrada de datos.

- **RF2.4:** métodos de autenticación efectivos
  - Otra de las características de la usabilidad es la efectividad. Los métodos implementados no deben generar errores a la hora de autenticarse, para que no haya que realizar más intentos de los necesarios.

- **RF2.5:** resolución mínima
  - Puesto que las imágenes para las contraseñas visuales necesitan un tamaño mínimo para poder pulsar en la zona correcta, la aplicación solo será ejecutable en dispositivos que permitan este tamaño mínimo.

### 4.2.3 Requisitos no Funcionales

En este aparatado presentaremos los requisitos no funcionales, es decir, aquellos que hacer referencia a aspectos de la aplicación no relacionados con la funcionalidad.

Se definen los siguientes requisitos no funcionales:

- **RNF1:** disponibilidad de varios idiomas
  - Android ofrece la posibilidad de añadir idiomas a sus aplicaciones de forma sencilla. Por ello la aplicación estará en inglés o español según la configuración del teléfono.

- **RNF2:** modularidad
    - Con el fin de poder reutilizar código para otras aplicaciones donde sea necesario un sistema de autenticación, se realizará un diseño modular para facilitar la reutilización.

- **RNF3:** escalabilidad
    - En línea con el RNF2, la aplicación se diseñará para poder integrar otros sistemas de autenticación de manera sencilla.

- **RNF4:** documentación
    - Todo el código deberá estar debidamente comentado y organizado para facilitar su mantenimiento o integración con otros sistemas.

## 4.3 Estructura del proyecto

En este punto explicaremos la estructura del proyecto, cómo están estructuradas las clases y los módulos.

### 4.3.1 Estructura del código

Con el fin de establecer una estructura modular para poder implementar nuevos sistemas de manera sencilla o para implementar estos sistemas en otras aplicaciones, hemos divido las actividades y clases en paquetes según su función y el sistema que implementen.

La organización que se ha seguido es la división de las clases en paquetes nombrados como “aplicación.tipo.paquete”. A continuación se detallan los paquetes que forman la aplicación y las clases que contienen:

- **tfg.activity.ccp:** este paquete contiene las actividades que implementan el sistema de autenticación mediante una contraseña visual CCP.
    - **AccountCCP:** actividad que permite a los usuario crear un usuario.
    - **LoginCCP**: actividad que permite autenticar a los usuario.

- **tfg.activity.menus:** este paquete contiene las actividades que sirven como menús para la navegación entre los distintos sistemas de autenticación.
    - **MenuInicio:** actividad inicial, nos permite elegir entre crear un usuario o autenticarse. También permite borrar la base de datos y guardar el log de creación y autenticación de usuarios.
    - **MenuLogin:** menú para elegir el sistema que queremos utilizar para autenticar un usuario.
    - **MenuAccount:** menú para elegir el sistema que queremos utilizar para crear un usuario.

- **tfg.activity.password:** este paquete contiene las actividades que implementan el sistema de autenticación mediante una contraseña alfanumérica.

  - **AccountPassword:** actividad que permite crear un usuario
  - **LoginPassword:** actividad que permite autenticar a un usuario

- **tfg.activity.tapi:** este paquete contiene las actividades que implementan el sistema de autenticación mediante una contraseña visual Tapi.
  - **AccountTapi:** actividad que permite crear un usuario.
  - **LoginTapi:** actividad que permite autenticar a un usuario.

- **tfg.clases.db:** este paquete contiene la clase que implementa la base de datos.
  - **DatabaseAdapter:** clase que implementa la base de datos usando SQLite.

- **tfg.view.imagenCCP:** este paquete contiene la vista que hemos creado para poder implementar la imagen que se usa en el sistema de autenticación mediante contraseña visual CCP.
  - **ImagenViewCCP:** clase que extiende View que implementa la vista utilizada

- **Tfg.view.imagenTapi:** este paquete contiene la vista que hemos creado para poder implementar la imagen que se usa en el sistema de autenticación mediante contraseña visual Tapi.
  - **ImagenViewTapi:** clase que extiende View que implementa la vista utilizada.

### 4.3.2 Modelo Vista Controlador

El modelo vista controlador es un patrón de arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones [27]. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. Tal y como hemos definido en los requisitos, este patrón se ajusta a nuestras necesidades.

Este patrón está compuesto por tres componentes:

- **Modelo:** es la representación de la información con la cual el sistema opera y la lógica para operar con esa información. En nuestro caso, un sistema de autenticación mediante contraseñas visuales, la información con la que se trabaja son los nombres de usuario y las contraseñas. De esta forma nuestro modelo es la base de datos que hemos implementado con SQlite ya que se encarga de realizar toda la lógica operacional de crear o autenticar usuarios.

- **Controlador:** responde a eventos e invoca peticiones al modelo cuando se hace una solicitud sobre la información que maneja. Y también puede enviar comandos a la vista para realizar algún cambio acorde al modelo. En nuestro caso, el controlador hace las peticiones a la base de datos para crear o autenticar usuarios. Las clases que implementan el papel de controlador son las actividades Login y Account para los distintos sistemas de autenticación.

- **Vista:** la vista es una representación gráfica del modelo. En nuestro caso la vista es una representación gráfica de la contraseña en nuestro sistema de autenticación. Las clases que implementan el papel de vista son las clases ImagenViewCCP e ImagenViewTapi.

### 4.3.3 Diagrama de clases

En este apartado podemos ver de forma gráfica lo explicado en el apartado anterior viendo cómo se relacionan las distintas clases para los distintos sistemas de autenticación:

- **Sistema de autenticación Password:** para este sistema no es necesario implementar una representación gráfica de la contraseña ya que esta se introduce por teclado y no mediante una imagen.

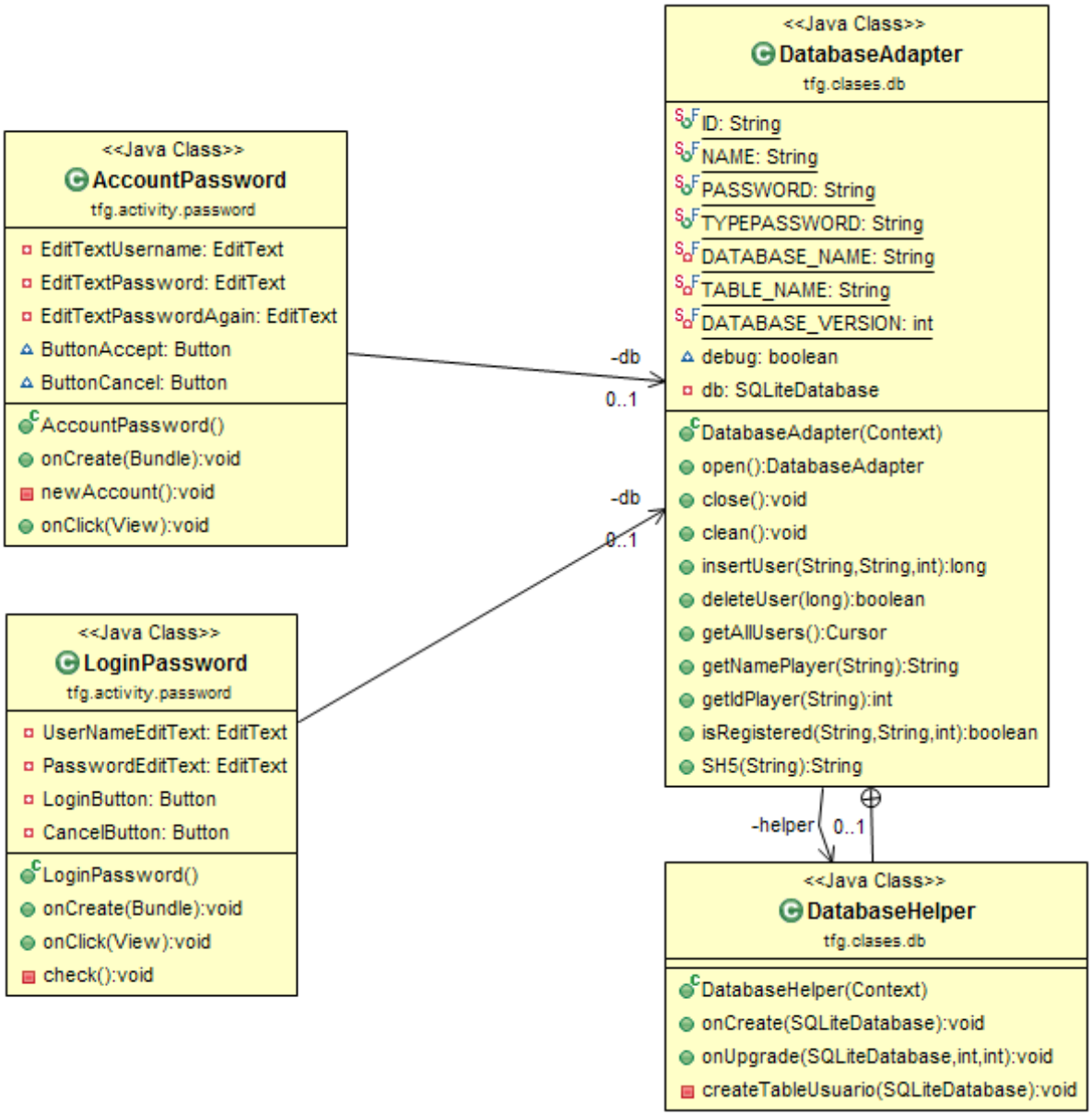


*Ilustración 14: Diagrama de clases para el módulo Password*

- **Sistema de autenticación TAPI:** las clases controladoras tienen acceso al modelo y a la vista.

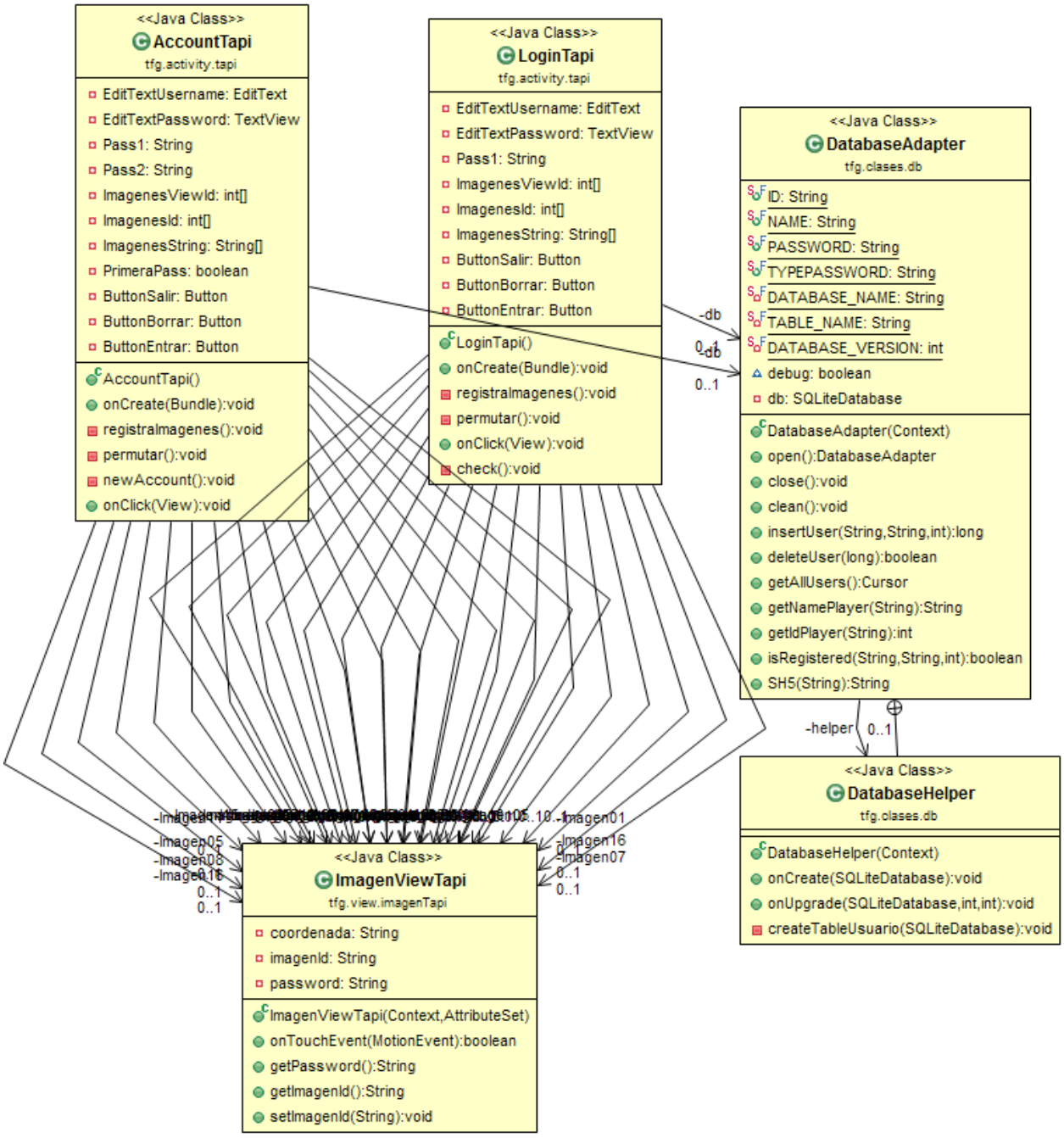


*Ilustración 15: Diagrama de clases para el módulo Tapi*

- **Sistema de autenticación CCP:** al igual que el anterior sistema, se puede observar la relación modelo vista controlador en el sistema CCP.

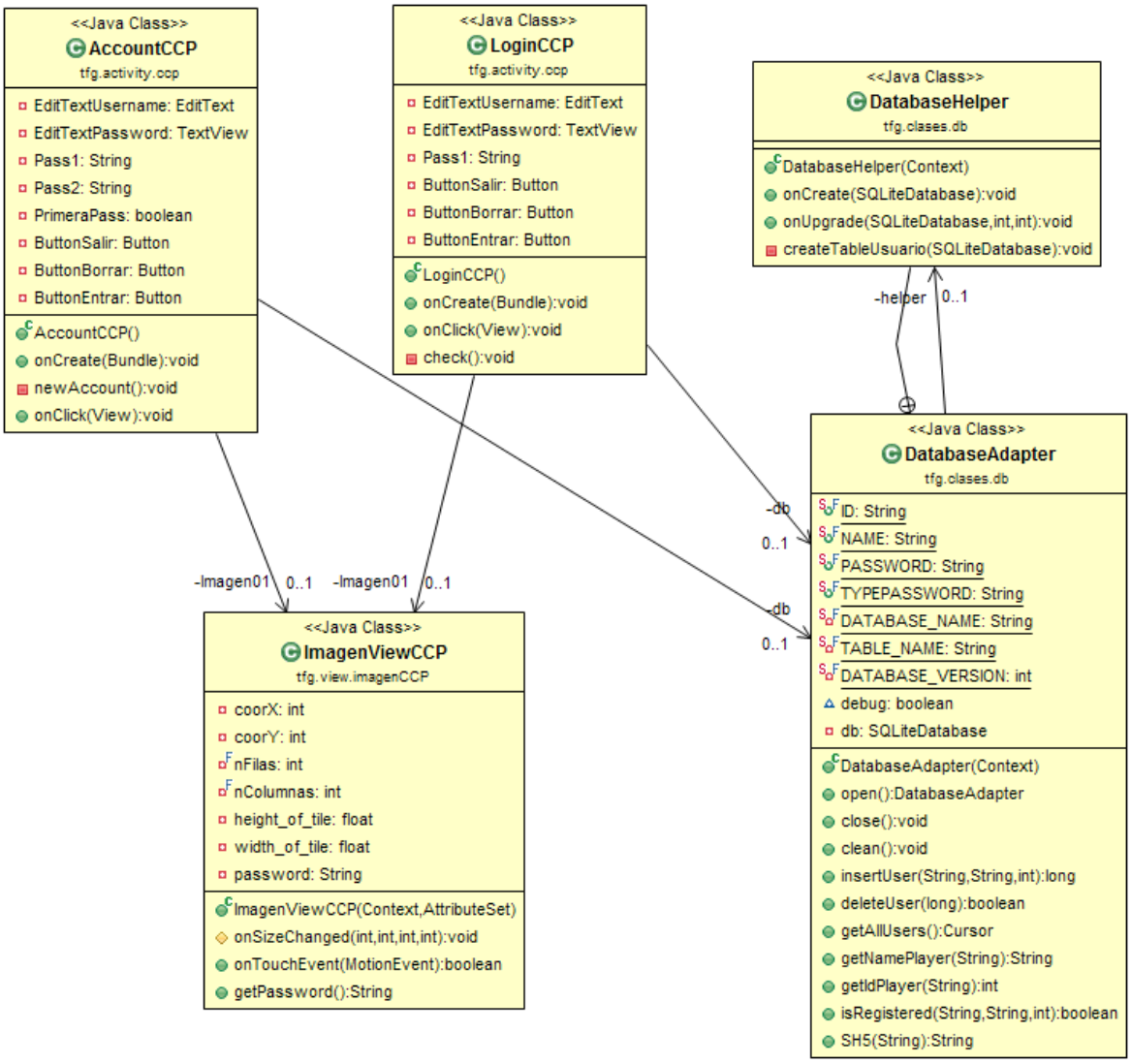


*Ilustración 16: Diagrama de clases para el módulo CCP*

## 4.4 Algoritmos implementados

En este punto explicaremos los algoritmos y sistemas de autenticación implementados y las decisiones tomadas en su diseño.

### 4.4.1 Base de datos

Android utiliza el motor de bases de datos transaccional SQLite que no precisa un servidor separado. Con SQLite, la base de datos es un simple fichero en el sistema de archivos Android:

*/data/data/package_name/databases*

Para el manejo de esta base de datos hemos creado la clase **DatabaseAdapter** donde se define la clase privada **DatabaseHelper** que es una subclase de **SQLiteOpenHelper**. El objetivo

de esta clase es crear la base de datos y actualizarla cuando sea necesario. Al extender la clase **SQLiteOpenHelper** hemos de crear las funciones **onCreate()** y **onUpgarde()**.

Cuando se llama por primera vez al creador de la clase **DatabaseHelper** crea la base de datos y es en esta clase donde se tienen que crear la estructura de tablas que forman nuestra base de datos. En nuestro sistema únicamente necesitamos una tabla de usuarios donde guardar el nombre y la contraseña, puesto que implementamos varios sistemas hemos añadido un campo identificativo para saber qué sistema de autenticación ha sido utilizado.

La clase **DatabaseAdapter** es la encargada de realizar las consultas a la base de datos. Para ello antes de poder realizar cualquier consulta es necesario de llamar a la función **open()** que invoca al método **getWritableDatabase()** que devuelve una referencia de tipo **SQLiteDatabase** para modificar los datos.

Una vez "abierta" la base de datos, podemos realizar las consultas necesarias para insertar, leer o borrar registros en la base de datos. Pero una ver realizadas todas consultas se debe llamar a la función **close()** para cerrarla, ya que si no se cierran tendríamos varias conexiones abiertas a la base de datos y eso podría crear incongruencias.

Puesto que nuestra aplicación es un sistema de autenticación debemos implementar funciones para insertar usuarios y comprobar si los usuarios ya están registrados: **isResgistered()** y **insertUser().**

Y como es un sistema que almacenan usuarios y contraseñas, ciframos las contraseñas antes de insertarlas en la base de datos con la función hash SH5. Android ofrece, de forma muy sencilla, la creación de este tipo de funciones con pocas sentencias, ya que no hace falta crear la función sino crear una instancia de ella.

En el anexo A se puede ver el código de esta clase.

### 4.4.2 Contraseña Alfanumérica

Este sistema se ha implementado para poder comparar la usabilidad con respecto a las contraseñas visuales. Para aumentar el espacio de claves y cumplir con lo especificado en los requisitos funcionales, no se hace ninguna restricción en la entrada de caracteres. Se permiten caracteres especiales y se hace diferencia entre mayúsculas y minúsculas, esto no es del todo real ya que en la mayora de sistemas que implementan sistemas de autenticación alfanumérica restringen la entrada de valores a letras, números y unos pocos caracteres especiales y muy pocos usuarios usan caracteres especiales en sus contraseñas.

Este módulo consta de dos actividades:

- **AccountPassword:** esta actividad es la encargada de registrar a los usuarios. Está formada por tres entradas de texto para el usuario, la contraseña y para repetir la contraseña. Además consta dos botones: uno para volver a menú anterior y otro para registrarse.

En caso de intentar registrarse sin introducir alguno de los valores, aparece un mensaje indicando que falta un valor por introducir.

En caso crear el usuario correctamente, volverá al menú anterior y aparecerá un mensaje indicando que se ha añadió el nuevo usuario.

En caso de intentar un usuario con el mismo nombre que uno existente, volverá al menú anterior y aparecerá un mensaje indicando que ya existe un usuario con ese nombre.

- **LoginPasword:** esta actividad es la encargada de autenticar a los usuarios que se hayan registrado usando este sistema. Está formada por 2 entradas de texto: una para el usuario y otra para la contraseña, y dos botones: uno para autenticarse y otro para volver al menú anterior.

  En caso de una autenticación exitosa, aparece un mensaje en la pantalla indicado que el usuario y la contraseña son correctos.

  En caso de dar error en la autenticación, aparece un mensaje en la pantalla indicando que el usuario o la contraseña son incorrectos.

### 4.4.3 Contraseña visual TAPI

En la versión implementada de la contraseña visual TAPI tenemos un conjunto de 16 imágenes, cada imagen se divide a su vez en 4 partes. El usuario cuando elige su contraseña debe acordarse del orden de las imágenes elegidas y el orden de las partes de estas imágenes. Las imágenes cambian de posición de forma aleatoria cada vez que se intenta acceder al sistema. De esta forma el espacio teórico de claves de este sistema es $64^n \times \frac{16!}{(16-p)!}$, donde *n* es la longitud de la contraseña y *p* el número de imágenes elegidas.

Al igual que en la contraseña alfanumérica, el módulo que implementa este sistema está divido en dos actividades: una para crear el usuario y otra para autenticarse. También hemos creado una subclase de **ImagenView** para recoger de forma más sencilla las pulsaciones en las imágenes.

- **ImagenViewTapi:** al ser una subclase de **ImagenView** hemos sobrecargado el método **onTouchEvent()** para poder recoger de forma correcta la posición que se ha pulsado dentro de la imagen. Además hemos definido 3 variables que nos ayudan a crear la contraseña: **coordenada**, **imagenId** y **contraseña**. La variable coordenada es posición en la que se ha pulsado, la variable **imagenId** es un identificador único y necesario ya que en este sistema contamos con 16 imágenes y por último la variable contraseña es donde almacenamos la contraseña generada al pulsar sobre la imagen. La contraseña es la concatenación de la variable **imagenId** y la variable **coordenada**.

- **AccountTapi:** esta actividad es la encargada de registrar a los usuarios mediante una contraseña visual TAPI. La actividad está formada por una entrada de texto donde introducir el nombre de usuario, 16 **ImagenesViewTapi** (definidas en el

punto anterior) y 3 botones para volver al menú anterior, borrar los datos introducidos por si el usuario se ha equivocado al introducir los valores y para registrarse. La primera vez que el usuario introduce la contraseña, el botón de registrarse pone repetir para así volver a introducir la contraseña y confirmarla.

Para colocar de forma aleatoria las imágenes, primero se asocian en orden las **ImagenViewTapi** de la actividad a las definidas en AccountTapi.xml, después se permuta de forma aleatoria el array que las contiene en la actividad. Una vez descolocadas todas las **ImagenViewTapi**, se las asocia una imagen y un id.

- **LoginTapi:** esta actividad es la encargada de autenticar a los usuarios que han sido creados con la actividad anterior. Tiene los mismos componentes y la lógica funcional es la misma. La principal diferencia es que no se solicita introducir por segunda vez la contraseña y que el botón que en la actividad anterior se usaba para registrar, ahora llama a la función que comprueba si el usuario existe.

### 4.4.4 Contraseña visual CCP

En la versión implementada de la contraseña visual CCP, tenemos una imagen que ocupa la mayor parte de la pantalla. El usuario cuando elige su contraseña debe pulsar en las zonas que quiera de la imagen. Cuando vaya a autenticarse deberá acordarse del orden y de las zonas que pulso cuando creó la contraseña. Este sistema utiliza una cuadricula que divide la imagen, esta división es necesaria ya que si la contraseña se generase con el pixel que ha pulsado el usuario, nunca conseguiría volver a pulsar otra vez en el mismo pixel cuando se quisiera autenticar. Esta cuadricula agrupa los pixel dividiendo la imagen en zonas y cada zona es una posible entrada para la contraseña. De esta forma el tamaño teórico de claves depende del tamaño de esta cuadricula. Pero hay que tener en cuenta que si los cuadrados son muy pequeños es muy posible que cuando se pulse en la pantalla se pulsen varios a la vez. Por otro lado si los cuadrados son muy grandes, es probable que zonas alejadas en la imagen formen parte del mismo cuadrado, provocando que disminuya la seguridad.

Si la imagen utilizada ocupa un tamaño de 1050x1180 px, en el caso de usar el ratón de un ordenador, podríamos dividir la imagen en cuadrados de hasta 19x19 px dando lugar a un espacio teórico de claves de $3410^n$, donde *n* es el número de pulsaciones, este número sale de realizar la siguiente operación $\frac{1050}{19} \times \frac{1180}{19}$. Pero al estar en un dispositivo móvil, las pulsaciones se realizan en la mayoría de los casos con el dedo y por ello hay que dividir la imagen en cuadrados de mínimo 100x100 px. Esto nos reduce el espacio de claves alrededor de $100^n$, donde *n* es número de pulsaciones.

Al igual que en las otras contraseñas, el módulo que implementa este sistema está divido en dos actividades: una para crear el usuario y otra para autenticarse. También hemos creado una subclase de **ImagenView** para recoger de forma más sencilla las pulsaciones en las imágenes.

- **ImagenViewCCP:** al ser una subclase de **ImagenView** hemos sobrecargado el método **onTouchEvent()** para poder recoger de forma correcta la posición que se ha pulsado dentro de la imagen. Para saber cuál ha sido el cuadrado pulsado, primero hay que saber el tamaño de cada cuadrado, de forma que cuando sabemos el pixel que se ha pulsado, se divide entre el ancho y alto del cuadrado para saber su posición en la cuadricula. La contraseña es la concatenación de la variable **coorX** y la variable **coorY**.

- **AccountCCP:** esta actividad es la encargada de registrar a los usuarios mediante una contraseña visual CCP. La actividad está formada por una entrada de texto donde introducir el nombre de usuario, una **ImagenViewCCP** (definida en el punto anterior) y 3 botones para volver al menú anterior, borrar los datos introducidos por si el usuario se ha equivocado al introducir los valores y para registrarse. La primera vez que el usuario introduce la contraseña, el botón de registrarse pone repetir para así volver a introducir la contraseña y confirmarla.

- **LoginTapi:** esta actividad es la encargada de autenticar a los usuarios que han sido creados con la actividad anterior. Tiene los mismos componentes y la lógica funcional es la misma. La principal diferencia es que no se solicita introducir por segunda vez la contraseña y que el botón que en la actividad anterior se usaba para registrar, ahora llama a la función que comprueba si el usuario existe.

## 4.5 Plan de Pruebas

Una vez desarrollado el proyecto e implementado los sistemas de autenticación, hemos procedido a realizar las pruebas necesarias para comprobar el funcionamiento y aceptación de los sistemas de autenticación por parte de los usuarios.

Al tratarse de una aplicación Android es difícil simular las posibles entradas que pueda generar un usuario al intentar registrarse usando una contraseña visual [28]. Por esta razón hemos decido dividir las pruebas de la siguiente forma:

- **Pruebas Unitarias y de Integración:** en estas pruebas hemos querido comprobar la correcta creación de los objetos en las actividades y la navegación entre ellas.

- **Pruebas con Usuarios:** para comprobar realmente la usabilidad de estos sistemas, hemos creído necesario realizar directamente pruebas con usuarios reales y no simulados.

### 4.5.1 Pruebas Unitarias y de Integración

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Con ello conseguimos asegurarnos que cada módulo funciona correctamente por separado. Una vez realizadas las pruebas de los módulos por separado y comprobado su funcionamiento, hay que realizar las pruebas de integración para comprar el comportamiento de todo el sistema.

Las pruebas unitarias deben cumplir los siguientes requisitos:

- **Automatizable:** no deben requerir una intervención manual.
- **Completas:** deben cubrir la mayor cantidad de código.
- **Repetibles:** no se deben crear pruebas que solo puedan ser ejecutadas una sola vez.
- **Independientes:** la ejecución de una prueba no debe interferir con la ejecución de otra.
- **Profesionales:** las pruebas deben ser consideradas parte del código, por tanto deben estar debidamente comentadas y documentadas.

#### *4.5.1.1 Robolectric*

Robolectric es un framework que simula la interfaz gráfica de un emulador de Android permitiéndonos ejecutar test en la propia máquina virtual de Java y ahorrando la necesidad de realizar los test en el emulador [29], [30].

Los test que se realizan con Robolectric son test normales de Junit 4, la única diferencia es que hay que incluir la anotación @RunWith(RobolectricTestRunner.class) antes de la declaración de la clase. Con esta anotación conseguimos realizar los test sin necesidad de levantar ningún emulador, con lo que ahorramos mucho tiempo de ejecución en los test. Otra de las ventajas de Robolectric es que permite aplicar TDD a nuestros desarrollos con Android, ya que podemos implementar casos de uso que interactúen con la interfaz gráfica sin que nuestros test provoquen fallos en la ejecución. También permite ejecutar los test sin necesidad de definir el *Instrumentation TestRunnner* en el archivo AndroidManifest ni tener que utilizar la librería de test de Android que solo funciona con un simulador funcionando.

#### *4.5.1.2 Pruebas realizadas*

Las pruebas realizadas para todas las actividades que forman la aplicación siguen la misma estructura:

- El primer test prueba que la actividad se ha creado correctamente y no es un objeto nulo.
- El segundo test prueba que los objetos que forman la actividad se han creado correctamente y no son objetos nulos.
- El tercer test prueba que los textos que hay en los objetos como Button o TextView son los que deben ser.
- El resto de los test realizados prueban la navegación de las actividades simulando una pulsación en los botones que permiten esta navegación.

En el anexo B se puede ver el resultado de todas las pruebas realizadas a cada una de las actividades. Y en el anexo C un código de ejemplo para ver la notación que usa Robolectric.

### 4.5.2 Pruebas de usuario

En todo sistema informático es importante realizar pruebas con usuarios ya que al final son ellos quienes decidirán si les gusta el sistema y si lo utilizarían. Las pruebas unitarias y de integración comprueban que no hay errores en la programación y que el sistema funciona correctamente. Pero las pruebas con usuarios pueden determinar el éxito o fracaso de una aplicación independientemente de lo bien o mal que este programada.

Para realizar las pruebas con usuarios, se les ha pedido que se cree un usuario y que un tiempo después traten de autenticarse. Para poder llevar un seguimiento de los resultados y poder compararlos después, hemos creado un log a partir de Logcat de Android.

Una de las razones de crear un log es que son una fuente para las auditorias. Los log son un historial que informa que fue cambiando y cómo fue cambiando, se recogen las modificaciones de datos y describe detalladamente la actividad general. Los log de auditoria que se utilizan en los sistemas y aplicaciones son el principal instrumento para detectar, diagnosticar, auditar y diagnosticar problemas de todo tipo.

Es importante definir lo que se quiere controlar ya que para que un log pueda ser útil, debe haberse realizado una labor previa de estudio de la información que los sistemas o aplicaciones pueden volcar en el log.

#### *4.5.2.1 LogCat*

Android ofrece un sistema log para recolectar y ver las salidas para depurar un sistema. Todos los logs de todas las aplicaciones y de parte del sistema se recogen en una serie de buffers circulares, a partir de los cuales se pueden ver y filtrar los mensajes con el comando 'logcat' en la ADB Shell.

Los mensajes que se crean en esta herramienta, comienzan con una etiqueta que informa del tipo de mensaje o error que ofrece:

- **V**: verbose (menos prioridad)
- **D**: debug
- **I**: información
- **W**: warning
- **E**: error
- **F**: fatal
- **S**: silent (mayor prioridad)

Siguiendo está notación, los mensajes que creamos para llevar un seguimiento de las actividades de los usuarios en nuestra aplicación llevan la etiqueta 'I' al tratarse de mensajes informativos.

#### *4.5.2.2 Pruebas realizadas*

Las pruebas se realizaron a lo largo de 2 semanas, para así comprobar si los usuarios eran capaces de recordar las contraseñas que habían utilizado en los distintos sistemas y para ver

cuántos intentos eran necesarios para poder crear un usuario y posteriormente cuantos intentos se necesitan para poder autenticarse con éxito.

En la siguiente tabla se puede observar el total de pruebas realizadas:

| | Password | | | TAPI | | | CCP | | |
|---|---|---|---|---|---|---|---|---|---|
| | Intentos | Éxitos | Fallos | Intentos | Éxitos | Fallos | Intentos | Éxitos | Fallos |
| **Registrarse** | 8 | 5 | 3 | 5 | 5 | 0 | 12 | 5 | 7 |
| **Autenticarse** | 16 | 10 | 6 | 19 | 8 | 11 | 23 | 9 | 14 |

*Tabla 2: Resultados de las pruebas con usuarios*

Si nos fijamos en los resultados, vemos que el sistema que menos problemas dio para crear usuarios, tuvo después una tasa alta de fallos a la hora de autenticar a los usuarios. Y el sistema CCP ha sido el que más fallos ha generado, tanto para registrar usuarios como para autenticarlos.

Los comentarios realizados por parte de los usuarios coinciden con lo que hemos defendido a lo largo de este documento:

- Resulta sencillo recordar una serie de imágenes.
- Es complicado acordarse de la parte pulsada en el sistema Tapi
- Es complicado pulsar dos veces en el mismo sitio en el sistema CCP.

En el anexo D se encuentra el log generado con las pruebas realizadas.

# 5. Conclusión y trabajo futuro

Como dijimos al principio de este documento, el principal objetivo de este proyecto era la de realizar un estudio sobre las ventajas e inconvenientes que tiene los sistemas de autenticación basados en contraseñas visuales frente a otros sistemas en dispositivos móviles Android.

Este estudio lo hemos realizado a lo largo de este documento, donde hemos analizado y explicado las características de los sistemas de seguridad: la confidencialidad, la integridad y la disponibilidad, las características de los sistemas de autenticación: seguridad y usabilidad. Una vez explicados las necesidades de seguridad y usabilidad de deben cumplir los sistemas de autenticación hemos procedido a explicar y detallar en qué consisten las contraseñas visuales y sus principales ventajas y desventajas. Y para comprobar su usabilidad también hemos creado una aplicación con la que hemos probado de forma práctica y con usuarios reales los sistemas de autenticación basados en contraseñas visuales en dispositivos Android.

Tras ello podemos concluir que las contraseñas visuales realmente cumplen ampliamente las necesidades de usabilidad y seguridad que deben tener los sistemas de autenticación. Aunque las pruebas de autenticación en la aplicación no fueron del todo satisfactorias, los comentarios hacía el sistema fueron positivos.

Con respecto al trabajo futuro en este campo, queda mucho por realizar ya que muy pocos sistemas de seguridad implementan este tipo de contraseñas. Pero creemos que los sistemas más seguros son aquellos que implementan varios procesos de autenticación, por ello pensamos que este tipo de contraseñas se deberían de implementar como otra forma de autenticación para aumentar la seguridad de los sistemas [31]. También es posible la implementación de las contraseñas visuales como si fuera una OTP en sistemas multicanal, por ejemplo cuando haces un pago por internet se manda un código al móvil para validar la transacción, en este caso se podría indicar una zona en la se debe pulsar en una imagen para validar la transacción.

Aunque en este estudio nos hemos centrado en dispositivos móviles Android, estos sistemas se pueden implementar del mismo modo en dispositivos móviles con otros sistemas operativos.

Desde un punto de vista más personal, este trabajo me ha servido para afianzar y ampliar los conocimientos adquiridos durante la carrera sobre seguridad y sistemas de autenticación al aplicarlos en un caso práctico. De la misma forma también he ampliado mis conocimientos de programación en Android.

# Glosario

**AES:** Advanced Encryption Standard, es un esquema de cifrado por bloques de 128 bits y usa tamaños de calve de 128, 192 o 256 bits.

**Android Developer Tools (ADT):** plugin para eclipse que lo provee de un entorno de desarrollo de nivel profesional para la creación de aplicaciones de Android. Es un completo IDE Java con funciones avanzadas para ayudar a crear, probar, depurar y empaquetar sus aplicaciones Android.

**APIs:** interfaz de programación de aplicaciones o Application Programming Interface

**ARM:** Advanced RISC Machine. Es una arquitectura RISC (Reduced Instruction Set Computer u Ordenador con Conjunto Reducido de Instrucciones) de 32 bits desarrollada por ARM Holdings.

**Backups:** copia de seguridad de los datos originales que se realiza con el fin de disponer de un medio de recuperarlos en caso de su pérdida.

**CCP:** sistema de contraseña visual que se basa en la memorización de una serie de puntos dentro de una imagen

**Dalvik:** máquina virtual que utiliza la plataforma para dispositivos móviles Android, permite ejecutar aplicaciones programadas en Java.

**DoS:** ataque de denegación de servicios, es un ataque a un sistema que causa que un servicio o recurso sea inaccesible a los usuarios legítimos.

**Framework:** en el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Puede incluir soporte programas, bibliotecas y un lenguaje interpretado.

**I.A.:** la inteligencia artificial es un área multidisciplinaria que, a través de la informática, la lógica y la filosofía estudia la creación y diseño de entidades capaces de razonar por si mismas utilizando como paradigma la inteligencia humana.

**IDE:** entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación. Es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

**Keylogger:** es un tipo de software o hardware que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un fichero a través de internet.

**MD4:** es un algoritmo de resumen del mensaje, implementa una función criptográfica hash para el uso de comprobaciones de integridad de mensajes.

**MD5:** es un algoritmo de resumen del mensaje, implementa una función criptográfica hash de 128 bits, fue desarrollado como remplazo del algoritmo MD4.

**One-Time Password (OTP):** es una contraseña válida solo para un uso en una autenticación.

**Open source:** es la expresión con la que se conoce al software distribuido y desarrollado libremente.

**QEMU:** Es un emulador de procesadores basado en la traducción dinámica de binarios. También tiene capacidades de virtualización dentro de un sistema operativo.

**RF:** requisito funcional

**RNF:** requisito no funcional

**ROMs:** en el contexto de Android, se refiere a las imágenes del sistema operativo que crean los usuarios con el fin de añadir sus propias modificaciones.

**Runtime:** se denomina runtime o tiempo de ejecución al intervalo de tiempo en el que una aplicación se ejecuta en un sistema operativo.

**SHA-1:** función hash que produce una salida de 160 bits de un mensaje que puede tener un tamaño de $2^64$ bits.

**Shoulder surfing:** técnica directa de observación para conseguir información como puedan ser contraseñas, pins o códigos de seguridad.

**Software Development Kit (SDK):** es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

**Spyware:** el spyware o programa espía es un software que recopila información de un ordenador y después transmite esta información a una entidad externa sin el conocimiento o el consentimiento del propietario del ordenador.

**TAPI:** sistema de autenticación de contraseñas visuales basado en el reconocimiento y memorización de claves en una imágenes.

**TDD:** desarrollo guiado por pruebas de software.

**Webkit:** es una plataforma para aplicaciones que funciona como base para navegadores web.

# Referencias


[1] D. A. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. .

[2] C. Shannon, "Communication theory of secrecy systems," *Bell Sys. Tech. J*, vol. 28, pp. 656–715, 1949.

[3] D. R. Stinson, *Cryptography: Theory and Practice*. .

[4] iso.org, "ISO27001: Information security management system (ISMS) standard." [Online]. Available: http://www.27000.org/iso-27001.htm, October 2005.

[5] iso.org, "ISO 7498-2:1989." [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256.

[6] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. 1997.

[7] Wikibooks, "Fundamentals of Information Systems Security/Access Control Systems." [Online]. Available: http://en.wikibooks.org/wiki/Fundamentals_of_Information_Systems_Security/Information_Security_and_Risk_Management#Core_Information_Security_Principles. [Accessed: 15-Apr-2014].

[8] L. F. Cranor and S. Garfinkel, *Introduction: Secure or Usable? Security & Privacy*. .

[9] A. Narayanan and V. Shmatikov, *Proceedings of the 12th ACM conference on Computer and communications security*. .

[10] E. Tittle and J. Stewart, *CISSP: Certified Information Systems Security Professional Study Guide*. 2006.

[11] iso.org, "ISO 9241-11:1998." [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=16883.

[12] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.," *IACR Cryptol. ePrint Arch.*, vol. 5, pp. 5–8, 2004.

[13] G. E. Blonder, "Graphical password," 24-Sep-1996.

[14] F. Schaub, M. Walch, B. Könings, and M. Weber, "Exploring the Design Space of Graphical Passwords on Smartphones," 2013.

[15] W. Z. Khan, M. Y. Aalsalem, and Y. Xiang, "A Graphical Password Based System for Small Mobile Devices," vol. 8, no. 5, pp. 145–154, 2011.

[16] S. Wiedenbeck and A. Brodskiy, "Authentication Using Graphical Passwords : Basic Results Background on Passwords Problems with Alphanumeric Passwords," 2001.

[17] G. S. Owen, “Graphical Passwords: A Survey,” *21st Annu. Comput. Secur. Appl. Conf.*, pp. 463–472.

[18] B. B. Zhu, D. Wei, M. Yang, and J. Yan, “Security Implications of Password Discretization for Click-based Graphical Passwords,” pp. 1581–1591, 2013.

[19] J. Chen, “The Comparison and Application of Corner Detection Algorithms,” vol. 4, no. 6, pp. 435–441, 2009.

[20] R. Collins and P. State, “Lecture 06 : Harris Corner Detector Motivation : Matchng Problem.”

[21] J. P. Gravel, “Corner Detection.”

[22] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Procedings Alvey Vis. Conf. 1988*, pp. 23.1–23.6, 1988.

[23] W. Moncur, “Pictures at the ATM : Exploring the usability of multiple graphical passwords,” 2007.

[24] M. P. D. Kulkarni, C. S. Satsangi, and S. Easo, “Authentication System for Banking Using Implicit Password,” vol. 3, no. 1, pp. 58–61, 2012.

[25] A. Vilchez, “Que es Android: Características y Aplicaciones.” [Online]. Available: http://www.configurarequipos.com/doc1107.html. [Accessed: 15-Apr-2014].

[26] S. Gunasekera, *Android Apps Security*. 2012.

[27] Wikipedia, “Modelo–vista–controlador.” [Online]. Available: http://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[28] N. De and A. De, “Infraestructura de Simulación Social para el Testado y Validación Previa al Despliegue de Aplicaciones Android Memoria del Trabajo Fin de Máster.”

[29] L. Vogel, “Using Robolectric for Android testing - Tutorial.” [Online]. Available: http://www.vogella.com/tutorials/Robolectric/article.html. [Accessed: 06-May-2014].

[30] robolectric, “robolectric.” [Online]. Available: http://robolectric.org/.

[31] P. C. Van Oorschot, “TwoStep : An Authentication Method Combining Text and Graphical Passwords,” no. 2.

# Anexo A:

```
private static class DatabaseHelper extends SQLiteOpenHelper {

	public DatabaseHelper(Context context) {
		super(context, DATABASE_NAME, null, DATABASE_VERSION);
	}


	@Override
	public void onCreate(SQLiteDatabase arg0) {
		createTableUsuario(arg0);
	}


	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		Log.w("DatabaseAdapter", "Upgrading database from version "
			+ oldVersion + " to version " + newVersion);
		db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
		createTableUsuario(db);
	}

	private void createTableUsuario(SQLiteDatabase db) {
		String str = "CREATE TABLE " + TABLE_NAME + " (" + ID
			+ " INTEGER PRIMARY KEY AUTOINCREMENT, " + NAME
			+ " TEXT NOT NULL, " + PASSWORD + " TEXT NOT NULL, "
			+ TYPEPASSWORD + " INTEGER);";
		try {
			db.execSQL(str);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}

public DatabaseAdapter open() throws SQLException {
	db = helper.getWritableDatabase();
	return this;
}

public void close() {
	helper.close();
}

public void clean() {
	open();
	helper.onUpgrade(db, DATABASE_VERSION, DATABASE_VERSION);
	close();
}

public long insertUser(String username, String password, int type) {
	String[] args = new String[] { username };
	Cursor c = db.rawQuery(" SELECT _id FROM users WHERE username=?", args);
	if (c.moveToFirst())
		return -1;

	ContentValues values = new ContentValues();
	values.put(NAME, username);
	values.put(PASSWORD, SH5(password));
	values.put(TYPEPASSWORD, type);
```

```
	return db.insert(TABLE_NAME, null, values);
}

public boolean isRegistered(String username, String password, int type) {
	boolean in = false;
	Cursor cursor = db.query(TABLE_NAME, new String[] { NAME, PASSWORD,
			TYPEPASSWORD }, NAME + " = '" + username + "' AND " +
			PASSWORD + "= '" + SH5(password) + "' AND " + TYPEPASSWORD +
			"= '" + type + "'", null, null, null, NAME + " DESC");
	if (cursor.moveToFirst())
		in = true;
	if (!cursor.isClosed())
		cursor.close();
	return in;
}

public String SH5(String s) {
	try {
		// Create SH5 Hash
		MessageDigest digest = java.security.MessageDigest
				.getInstance("SHA-512");
		digest.update(s.getBytes());
		byte messageDigest[] = digest.digest();

		// Create Hex String
		StringBuffer hexString = new StringBuffer();
		for (int i = 0; i < messageDigest.length; i++)
			hexString.append(Integer.toHexString(0xFF &
			messageDigest[i]));
			return hexString.toString();

	} catch (NoSuchAlgorithmException e) {
		e.printStackTrace();
	}

	return "";
	}
}
```

# Anexo B:

```
<?xml version="1.0" encoding="UTF-8"?>
- <testrun ignored="0" errors="0" failures="0" started="41" tests="41" project="TFGElectricTest" name="TFGElectricTest">
  - <testsuite name="tfg.test.login.TestLoginTapi" time="11.716">
      <testcase name="TestActivityIsNotNull" time="8.901" classname="tfg.test.login.TestLoginTapi"/>
      <testcase name="TestIsNotNull" time="0.933" classname="tfg.test.login.TestLoginTapi"/>
      <testcase name="TestTextButton" time="0.968" classname="tfg.test.login.TestLoginTapi"/>
      <testcase name="TestClikAtras" time="0.914" classname="tfg.test.login.TestLoginTapi"/>
    </testsuite>
  - <testsuite name="tfg.test.account.TestAccountCCP" time="3.513">
      <testcase name="TestActivityIsNotNull" time="0.892" classname="tfg.test.account.TestAccountCCP"/>
      <testcase name="TestIsNotNull" time="0.868" classname="tfg.test.account.TestAccountCCP"/>
      <testcase name="TestTextButton" time="0.882" classname="tfg.test.account.TestAccountCCP"/>
      <testcase name="TestClikAtras" time="0.871" classname="tfg.test.account.TestAccountCCP"/>
    </testsuite>
  - <testsuite name="tfg.test.menu.TestMenuInicio" time="2.446">
      <testcase name="TestActivityIsNotNull" time="0.454" classname="tfg.test.menu.TestMenuInicio"/>
      <testcase name="TestTextViewIsNotNull" time="0.448" classname="tfg.test.menu.TestMenuInicio"/>
      <testcase name="TestTextMenuInicio" time="0.645" classname="tfg.test.menu.TestMenuInicio"/>
      <testcase name="TestClikMenuRegistro" time="0.452" classname="tfg.test.menu.TestMenuInicio"/>
      <testcase name="TestClikMenuLogin" time="0.447" classname="tfg.test.menu.TestMenuInicio"/>
    </testsuite>
  - <testsuite name="tfg.test.account.TestAccountTapi" time="3.82">
      <testcase name="TestActivityIsNotNull" time="0.988" classname="tfg.test.account.TestAccountTapi"/>
      <testcase name="TestIsNotNull" time="0.938" classname="tfg.test.account.TestAccountTapi"/>
      <testcase name="TestTextButton" time="0.951" classname="tfg.test.account.TestAccountTapi"/>
      <testcase name="TestClikAtras" time="0.943" classname="tfg.test.account.TestAccountTapi"/>
    </testsuite>
  - <testsuite name="tfg.test.menu.TestMenuLogin" time="2.335">
      <testcase name="TestActivityIsNotNull" time="0.431" classname="tfg.test.menu.TestMenuLogin"/>
      <testcase name="TestTextViewIsNotNull" time="0.374" classname="tfg.test.menu.TestMenuLogin"/>
      <testcase name="TestTextMenuInicio" time="0.368" classname="tfg.test.menu.TestMenuLogin"/>
      <testcase name="TestClikPassword" time="0.387" classname="tfg.test.menu.TestMenuLogin"/>
      <testcase name="TestClikTapi" time="0.394" classname="tfg.test.menu.TestMenuLogin"/>
      <testcase name="TestClikCCP" time="0.381" classname="tfg.test.menu.TestMenuLogin"/>
    </testsuite>
  - <testsuite name="tfg.test.login.TestLoginCCP" time="3.513">
      <testcase name="TestActivityIsNotNull" time="0.887" classname="tfg.test.login.TestLoginCCP"/>
      <testcase name="TestIsNotNull" time="0.882" classname="tfg.test.login.TestLoginCCP"/>
      <testcase name="TestTextButton" time="0.873" classname="tfg.test.login.TestLoginCCP"/>
      <testcase name="TestClikAtras" time="0.871" classname="tfg.test.login.TestLoginCCP"/>
    </testsuite>
  - <testsuite name="tfg.test.login.TestLoginPassword" time="3.434">
      <testcase name="TestActivityIsNotNull" time="0.95" classname="tfg.test.login.TestLoginPassword"/>
      <testcase name="TestTextButton" time="0.838" classname="tfg.test.login.TestLoginPassword"/>
      <testcase name="TestClikAtras" time="0.826" classname="tfg.test.login.TestLoginPassword"/>
      <testcase name="TestViewIsNotNull" time="0.82" classname="tfg.test.login.TestLoginPassword"/>
    </testsuite>
  - <testsuite name="tfg.test.account.TestAccountPassword" time="3.996">
      <testcase name="TestActivityIsNotNull" time="0.949" classname="tfg.test.account.TestAccountPassword"/>
      <testcase name="TestTextButton" time="0.922" classname="tfg.test.account.TestAccountPassword"/>
      <testcase name="TestClikAtras" time="0.925" classname="tfg.test.account.TestAccountPassword"/>
      <testcase name="TestViewIsNotNull" time="1.2" classname="tfg.test.account.TestAccountPassword"/>
    </testsuite>
  - <testsuite name="tfg.test.menu.TestMenuAccount" time="2.278">
      <testcase name="TestActivityIsNotNull" time="0.402" classname="tfg.test.menu.TestMenuAccount"/>
      <testcase name="TestTextViewIsNotNull" time="0.371" classname="tfg.test.menu.TestMenuAccount"/>
      <testcase name="TestTextMenuInicio" time="0.373" classname="tfg.test.menu.TestMenuAccount"/>
      <testcase name="TestClikPassword" time="0.372" classname="tfg.test.menu.TestMenuAccount"/>
      <testcase name="TestClikTapi" time="0.379" classname="tfg.test.menu.TestMenuAccount"/>
      <testcase name="TestClikCCP" time="0.38" classname="tfg.test.menu.TestMenuAccount"/>
    </testsuite>
</testrun>
```

# Anexo C:

```
package tfg.test.menu;

import static org.hamcrest.CoreMatchers.equalTo;
import static org.junit.Assert.*;

…

@RunWith(RobolectricTestRunner.class)
public class TestMenuInicio {

	private MenuInicio menuInicio;

	/*
	 * Iniciamos el objeto de este Test
	 */
	@Before
	public void setup(){
		menuInicio = Robolectric.buildActivity(MenuInicio.class)
				.create().get();
	}

	/*
	 * test para comprobar que la actividad no es NULL
	 */
	@Test
	public void TestActivityIsNotNull() throws Exception {

		assertNotNull(menuInicio);
	}

	/*
	 * Test para comprar que los objetos View no son NULL
	 */
	@Test
	public void TestTextViewIsNotNull() throws Exception {

	TextView textView2 = (TextView) menuInicio.findViewById(R.id.registro);
	assertNotNull(textView2);

	TextView textView3 = (TextView) menuInicio.findViewById(R.id.login);
	assertNotNull(textView3);

	TextView textView4 = (TextView) menuInicio.findViewById(R.id.clean);
		assertNotNull(textView4);
	}

	/*
	 * Test para comprobar el texto de los objetos View
	 */
	 @Test
	 public void TestTextMenuInicio() throws Exception {

	TextView textView2 = (TextView) menuInicio.findViewById(R.id.registro);
	String text2 = (String) textView2.getText();
	assertThat(text2,
	equalTo(menuInicio.getResources().getString(R.string.registrarse)));

	TextView textView3 = (TextView) menuInicio.findViewById(R.id.login);
	String text3 = (String) textView3.getText();
```

```
        assertThat(text3,
        equalTo(menuInicio.getResources().getString(R.string.login)));

        TextView textView4 = (TextView) menuInicio.findViewById(R.id.clean);
        String text4 = (String) textView4.getText();
               assertThat(text4,
        equalTo(menuInicio.getResources().getString(R.string.borrarDatos)));

        }

        /*
         * Test para comprobar la navegacion del menu
         */
         @Test
         public void TestClikMenuRegistro() throws Exception {

        TextView textView2 = (TextView) menuInicio.findViewById(R.id.registro);

        textView2.performClick();
        Intent intent2 =
        Robolectric.shadowOf(menuInicio).peekNextStartedActivity();
        assertEquals(MenuRegistro.class.getCanonicalName(), intent2.getAction());

         }

         /*
          * Test para comprobar la navegacion del menu
          */
         @Test
         public void TestClikMenuLogin() throws Exception {

        TextView textView3 = (TextView) menuInicio.findViewById(R.id.login);

        textView3.performClick();
        Intent intent3 =
        Robolectric.shadowOf(menuInicio).peekNextStartedActivity();
        assertEquals(MenuLogin.class.getCanonicalName(), intent3.getAction());

        }

}
```

# Anexo D:

```
--------- beginning of /dev/log/main
05-06 19:45:54.473 I/TFG     ( 1343): AccountPassword: prueba1 Usuario creado
05-06 19:46:12.273 I/TFG     ( 1343): AccountTapi: prueba2 Usuario creado
05-06 19:46:27.773 I/TFG     ( 1343): AccountCCP: prueba3 Usuario creado
05-06 19:46:42.083 I/TFG     ( 1343): LoginPassword: prueba1 Inicio de sesión correcto
05-06 19:46:54.263 I/TFG     ( 1343): LoginTapi: prueba2 Inicio de sesión correcto
05-06 19:47:04.143 I/TFG     ( 1343): LoginCCP: prueba3 Inicio de sesión correcto
05-07 09:51:58.846 I/TFG     (30298): AccountPassword: alvaro1 Fallo al crear usuario,
falta valor
05-07 09:52:08.606 I/TFG     (30298): AccountPassword: alvaro1 Usuario creado
05-07 09:52:47.426 I/TFG     (30298): AccountTapi: alvaro2 Usuario creado
05-07 09:53:22.516 I/TFG     (30298): AccountCCP: alvaro3 Fallo al crear usuario, no
coinciden contraseñas
05-07 09:53:32.896 I/TFG     (30298): AccountCCP: alvaro3 Fallo al crear usuario, no
coinciden contraseñas
05-07 09:53:54.576 I/TFG     (30298): AccountCCP: alvaro3 Fallo al crear usuario, no
coinciden contraseñas
05-07 09:54:07.326 I/TFG     (30298): AccountCCP: alvaro3 Fallo al crear usuario, no
coinciden contraseñas
05-07 09:54:22.726 I/TFG     (30298): AccountCCP: alvaro3 Usuario creado
--------- beginning of /dev/log/main
05-09 09:15:36.166 I/TFG     ( 9035): AccountPassword: mj Fallo al crear usuario, no
coinciden contraseñas
05-09 09:15:40.396 I/TFG     ( 9035): AccountPassword: mj Fallo al crear usuario, no
coinciden contraseñas
05-09 09:16:29.486 I/TFG     ( 9035): AccountPassword: mj Usuario creado
05-09 09:17:34.576 I/TFG     ( 9035): AccountTapi: mj2 Usuario creado
05-09 09:18:44.346 I/TFG     ( 9035): AccountCCP: mj Fallo al crear usuario, no
coinciden contraseñas
05-09 09:19:41.316 I/TFG     ( 9035): AccountCCP: mj3 Fallo al crear usuario, no
coinciden contraseñas
05-09 09:19:53.846 I/TFG     ( 9035): AccountCCP: mj3 Usuario creado
--------- beginning of /dev/log/main
05-09 09:25:11.696 I/TFG     ( 9035): LoginPassword: alvaro Inicio de sesión incorrecto
05-09 09:25:20.106 I/TFG     ( 9035): LoginPassword: alvaro1 Inicio de sesión correcto
05-09 09:25:55.576 I/TFG     ( 9035): LoginTapi: alvaro2 Inicio de sesión incorrecto
05-09 09:26:10.006 I/TFG     ( 9035): LoginTapi: alvaro2 Inicio de sesión incorrecto
05-09 09:26:12.046 I/TFG     ( 9035): LoginTapi: alvaro2 Inicio de sesión incorrecto
05-09 09:26:19.196 I/TFG     ( 9035): LoginTapi: alvaro2 Inicio de sesión correcto
05-09 09:26:49.196 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
05-09 09:27:12.546 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión correcto
--------- beginning of /dev/log/main
05-09 11:21:05.636 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
05-09 11:21:12.436 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
05-09 11:21:18.816 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
05-09 11:21:21.536 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
05-09 11:21:28.196 I/TFG     ( 9035): LoginCCP: alvaro3 Inicio de sesión incorrecto
--------- beginning of /dev/log/main
05-09 15:46:28.546 I/TFG     (26508): LoginPassword: prueba1 Inicio de sesión correcto
05-09 15:48:43.232 I/TFG     (26508): LoginTapi: prueba2 Inicio de sesión correcto
05-09 15:53:00.596 I/TFG     (26508): LoginCCP: prueba3 Inicio de sesión correcto
05-09 15:53:15.116 I/TFG     (26508): LoginCCP: prueba3 Inicio de sesión correcto
--------- beginning of /dev/log/main
05-12 15:28:38.476 I/TFG     (18385): LoginPassword: ana1 Inicio de sesión incorrecto
05-12 15:28:57.106 I/TFG     (18385): AccountPassword: ana1 Usuario creado
05-12 15:29:19.666 I/TFG     (18385): AccountTapi: ana2 Usuario creado
05-12 15:29:35.686 I/TFG     (18385): AccountCCP: ana3 Usuario creado
05-12 15:29:50.256 I/TFG     (18385): LoginPassword: prueba1 Inicio de sesión correcto
05-12 15:30:04.056 I/TFG     (18385): LoginTapi: prueba2 Inicio de sesión correcto
05-12 15:30:13.706 I/TFG     (18385): LoginCCP: prueba3 Inicio de sesión incorrecto
05-12 15:30:20.506 I/TFG     (18385): LoginCCP: prueba3 Inicio de sesión correcto
--------- beginning of /dev/log/main
05-12 18:17:12.076 I/TFG     ( 8202): LoginPassword: ana1 Inicio de sesión incorrecto
05-12 18:17:17.236 I/TFG     ( 8202): LoginPassword: ana1 Inicio de sesión correcto
05-12 18:17:28.596 I/TFG     ( 8202): LoginTapi: ana1 Inicio de sesión incorrecto
05-12 18:17:40.926 I/TFG     ( 8202): LoginTapi: ana1 Inicio de sesión incorrecto
05-12 18:17:59.056 I/TFG     ( 8202): LoginTapi: ana1 Inicio de sesión incorrecto
05-12 18:18:16.106 I/TFG     ( 8202): LoginTapi: ana1 Inicio de sesión incorrecto
05-12 18:18:29.916 I/TFG     ( 8202): LoginTapi: ana2 Inicio de sesión correcto
05-12 18:18:41.526 I/TFG     ( 8202): LoginCCP: ana3 Inicio de sesión correcto
--------- beginning of /dev/log/main
05-13 14:44:59.108 I/TFG     ( 8033): LoginPassword: mj Inicio de sesión incorrecto
05-13 14:45:13.938 I/TFG     ( 8033): LoginPassword: mj Inicio de sesión correcto
05-13 14:45:37.688 I/TFG     ( 8033): LoginPassword: mj1 Inicio de sesión incorrecto
```

# Anexo E:

Pantallas de los sistemas de autenticación creados:

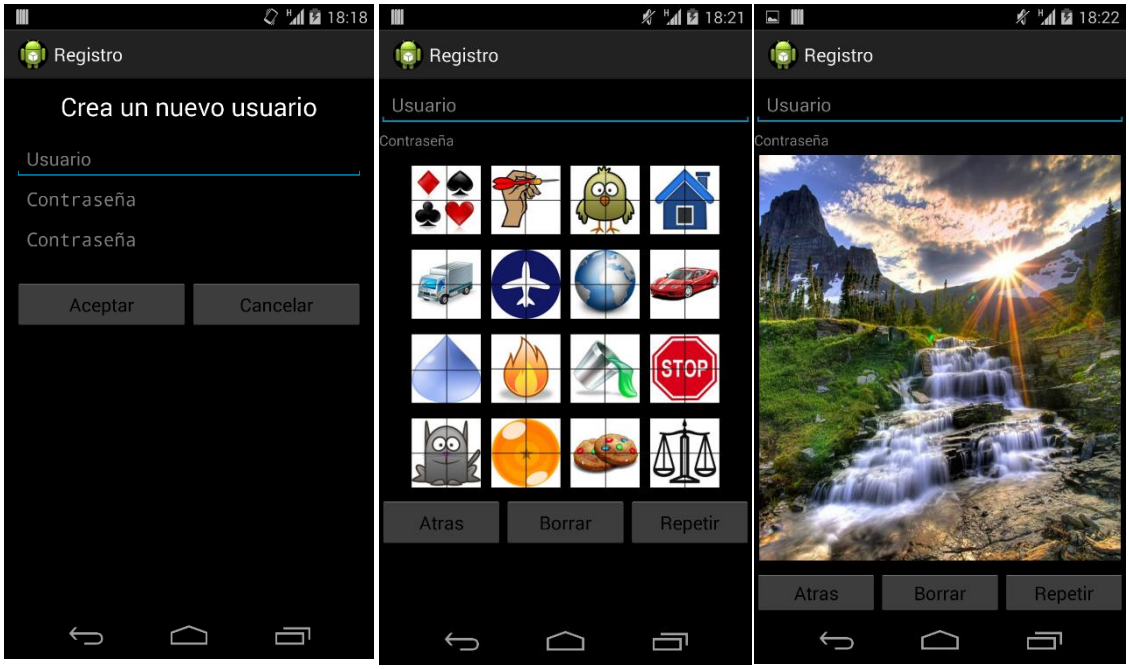


*Ilustración 17: capturas de la pantalla de Registro para los diferentes sistemas*

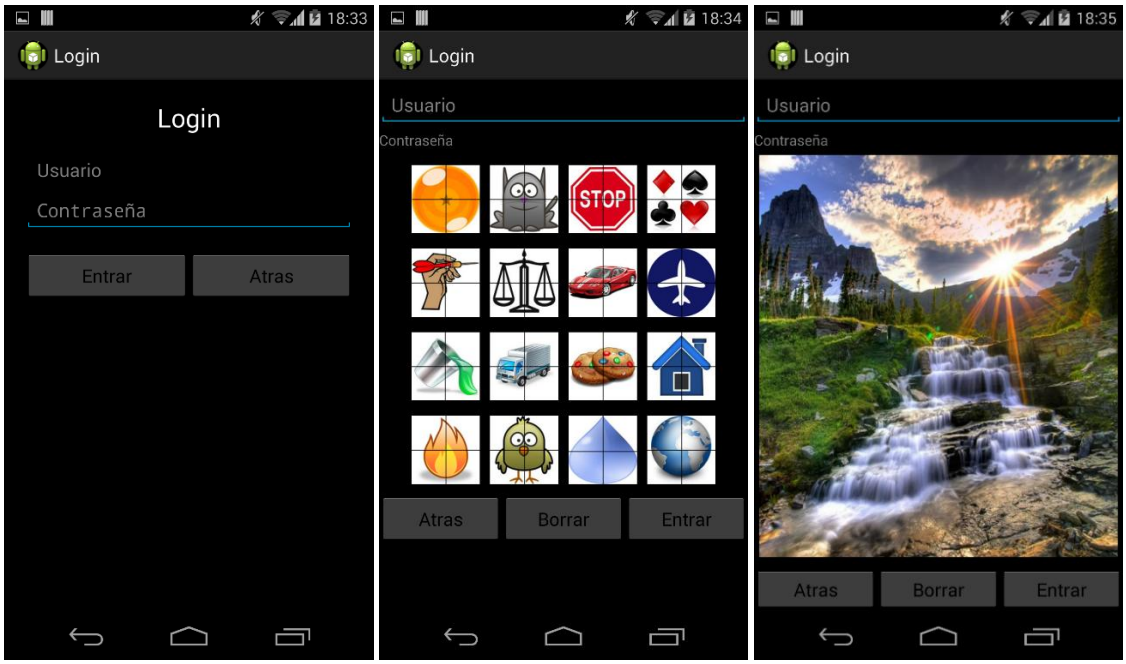


*Ilustración 18: Capturas de la pantalla de Autenticación para los diferentes sistemas*